MANGALORE UNIVERSITY

Final Synopsis

of the PhD Thesis to be submitted for the award of degree of
Doctor of Philosophy in

Computer Science

TITLE OF THE WORK

An Efficient Recommendation Systems for Social Media using Big Data Analytics

By

Mohammed Ahmed Mohammed Fadhel Aljunid
Research Scholar

Under the Guidance of
Dr. Manjaiah D.H,
Professor

Department of Post-Graduate Studies and Research in Computer Science,
Mangalore University,
Mangalagangotri-574 199

March 2022

Contents

1 Hypothesis and Research Issue **2**

1.1 Introduction . . . 2

1.2 Collaborative Filtering approaches . . . 3

1.3 Research Issue . . . 5

1.3.1 Cold-start problem . . . 5

1.3.2 Data sparsity problem . . . 5

1.3.3 Scalability problem . . . 5

1.4 Literature Review . . . 6

2 Objectives **10**

3 Proposed Models **10**

3.1 Efficient Hybrid Recommendation model Based on Collaborative Filtering Recommender Systems . . . 10

3.1.1 System Overview . . . 11

3.1.1.1 Problem Formulation . . . 11

3.1.1.2 Computed Similarity of UCF and ICF Models . . . 11

3.1.1.2.1 Cosine Similarity (COS) . . . 12

3.1.1.2.2 Pearson Correlation Coefficient (PCC) . . . 13

3.1.1.3 Rating Prediction of UCF and ICF Models . . . 13

3.1.1.4 Hybrid Proposed Model (ΓUICF) . . . 14

3.1.2 Experimental Study . . . 17

3.1.2.1 Experimental Setup . . . 17

3.1.2.2 Dataset Description . . . 17

3.1.2.3 Evaluation Metrics . . . 18

3.1.2.4 Modeling Setting . . . 18

3.1.2.5 Analysis and Discussion . . . 18

3.1.2.5.1 Baselines CF Model (BCF) . . . 19

3.1.2.5.2 Baseline UCF and ICF Models . . . 19

3.1.2.5.3 UCF and ICF with Baseline CF . . . 19

3.1.2.5.4 Topk UCF and Topk ICF Models . . . 20

3.1.2.5.5 Normalize UCF and ICF Models . . . 20

3.1.2.6 Results . . . 21

3.2 Integrating of Explicit and Implicit Feedback based on deep learning collaborative filtering algorithm . . . 24

3.2.1 Methodology Overview 24
3.2.1.1 Formulating the Problem 25
3.3 Coupling Formulation 25
3.3.0.1 Learning Explicit User-item Coupling 28
3.3.0.2 Learning Implicit User-item coupling 28
3.3.0.3 Fusion of Explicit and Implicit Feedback Couplings Models 30
3.3.0.3.1 Model Training 31
3.3.1 Experimental Study 32
3.3.1.1 Datasets Description 32
3.3.1.2 Evaluation Protocols 32
3.3.1.3 Parameter Setting 33
3.3.1.4 Time Complexity 33
3.3.1.5 Comparison with Baselines. 34
3.3.1.5.1 Our Models 34
3.3.1.5.2 Baseline Models 34
3.3.1.6 Results and Discussion 35
3.4 Movie Recommender System Based on Collaborative Filtering Using Apache Spark 37
3.4.1 Proposed Model 38
3.4.2 Experimental Results Analysis and Discussion 38

4 Conclusion and Future works **43**

5 Publications **44**

An Efficient Recommendation Systems for Social Media using Big Data Analytics

Abstract

Collaborative filtering (CF) algorithms are one of the most popular filtering techniques used in recommender systems for providing personalised services for users. The CF technique collects users' previous information about items such as books, music, movies, ideas, and so on and makes automatic predictions (filtering) about the interests of users. The CF algorithm has a shortcoming in the recommendation process, namely the cold start, sparsity, and scalability problems. Recently, research directions have been inclined toward considering and addressing the limitations of conventional CF by using machine learning and deep learning approaches due to their better accuracy in recommendation. To improve CF algorithms and meet the needs of recommendation system development, we must first analyse existing CF models and their limitations, and then make targeted improvements.

Firstly, we provide a survey of CF RS based on machine learning (ML) and deep learning (DL) algorithms to provide, enlighten, and guide beginner and advanced researchers interested in the domain of RS. It is divided into two dimensions: one for beginners and one for advanced readers. The first dimension is devoted to explaining the basic concept of an RS, while the second dimension is devoted to explaining the remedies for RS issues, tasks of each paper, most metrics, and most datasets among all studies. In this study, there are 278 systematic references that were chosen based on things like how many recent citations the article had and how important the journal, international conference, or book chapter where the article was published were.

And then, we propose a hybrid recommendation model, namely (ΓUICF). The ΓUICF model integrates user-based CF (UCF) and item-based CF (ICF) models with Γ linear regression to model the sparsity and scalability issues of the user-item rating matrix. The different major similarity computations between users to users and items to items are computed, and then the prediction for the unobserved target rating is made by adapting UCF and ICF models in the offline step. The Γ parameter linear regression model learns for each user and item and fuses the two predictions in the online step into a weighted model. The experiments were conducted on two different real-world datasets, namely the 100k MovieLens and 1M MovieLens datasets. The proposed model was observed to have significantly better performance when compared to the existing models.

Moreover, a novel neural recommendation model is proposed based on non-independent and identically distributed (Non-IID) for CF by incorporating explicit and implicit coupling interactions. The explicit interactions consist of two models, namely Intra-coupling interactions within users and items, and Inter-coupling interactions between different users and items concerning the attributes of users and items. The Intra-coupled model learns using deep learning convolutional neural

networks and is combined with the Inter-coupled model. Besides explicit coupling interactions, we present a Generalized Matrix Factorization Bias (GMFB) model that systematically trains the implicit user-item coupling. Finally, we combined explicit and implicit coupling interactions within and between users and items, accompanying the extra information about users and items under a framework called "IntegrateCF." Extensive experiments on two large real-world datasets have shown that the proposed model performs better than existing methods.

Finally, we propose a movie recommendation system based on (Alternating Least Squares) ALS model using Apache Spark. This research focuses on the selection of parameters of ALS algorithms that can affect the performance of a robust RS. From the results, a conclusion is drawn according to the selection of parameters of ALS algorithms, which can affect the performance of the building of a movie recommender engine. The model evaluation is done using different metrics such as execution time, root mean squared error (RMSE) of rating prediction, and the rank at which the best model was trained. Two best cases are chosen based on best parameter selection from experimental results, which can lead to building a good prediction rating for a movie recommender.

1 Hypothesis and Research Issue

The hypothesis of claims that, in the recommender systems, it is possible to develop a new recommendation model based on collaborative filtering recommender systems. This work suggests new recommendation models based on collaborative filtering for social media using big data analytic. Big data analysis such as machine learning and deep learning algorithms. In addition, big data tools such as Apache Spark, Hadoop, and so on.

1.1 Introduction

A recommender systems (RSs) is an information filtering algorithm, that makes suggestions for items that lead users based on their interest. In today's internet era, the RS plays a vital role in handling the information overload complexities [53]. With the swift growth of the internet and commercial enterprises, the size of data loaded to the internet rises alarmingly. The huge volume of data has led to information overload-related complexities on the internet. The RS has proven to be a successful and operative way to address the challenges associated with the information overload problem over the internet[27]. RSs are designed to augment human decision making. The formulation of recommendation problems is primarily designed to answer at least one of the two questions, namely the predicted version and the ranking version. The predicted version deals with predicting evaluation values for user-item connectivity.

The internal functions for RS are characterized by the filtering algorithm. The most widely used classification divides the filtering algorithms as content-based, collaborative filtering and hybrid approaches. Collaborative filtering recommendation technology is one of the earliest and most successful technologies used in recommendation systems [27]. It generally uses the nearest neighbor technology to calculate the distance between users using the user's historical preference information, and then uses the weighted evaluation value of the target user's nearest neighbor user's product evaluation to predict the target user's preference for a specific product. The system then uses this Recommend the target

user based on the degree of preference [23, 24, 46]. The advantage of CF is ability to effectively use the feedback information of other similar users, reduce the amount of user feedback, and accelerate the speed of personalized learning and ability to filter information that is difficult for automatic machine content analysis, such as artwork, music, etc 3). Another advantage is ability to recommend new information. However it has limitations such as cold start, data sparsity, scalability issues [90].

Content-based [34, 53, 77] defined by the characteristic attributes of the relevant content. The system learns the user's interest based on the characteristics of the user's evaluation object, and examines the matching degree of the user's profile with the item to be predicted. The user's profile model depends on the learning method used. Commonly used are decision trees, neural networks, and vector-based representation methods. Content-based user profile requires historical data of the user, and the user profile model may change as the user's preferences change [6, 20, 56]. The advantage of content-based is no cold start with new items and data sparse issues, its able to recommend users with special hobbies. However, the recommended content must be able to abstract meaningful features and have a good structure.

Hybrid approaches [45, 68, 83] is a recommendations on current web sites often do not simply use a certain recommendation mechanism and strategy. They often mix multiple methods to achieve better recommendation results.

In this work, we focus on collaborative filtering techniques based on recommendation system because it is a widely used algorithm in this field. Collaborative filtering technique was presented in the 1990s and has been successfully used in many recommender systems [9, 11]. It predicts a rating for a user based on the rating preferences of similar users. Most collaborative filtering algorithms operate by finding similar users and then predicting a rating of an item based on the preferences and previous ratings of those users. The datasets that are widely used in collaborative filtering technique is MovieLens dataset, which was created in 1997 by the GroupLens. GroupLens is a research lab in the Department of Computer Science and Engineering at the University of Minnesota. There are many significant fields in the lab, one of those is recommender systems some researcher works on this dataset. The improved correlation coefficient derives the higher accuracy in MovieLens datasets than others.

In this thesis, we presented many recommendation models which solved many problems of the collaborative filtering such as sparsity, scalability and cold start problem using advanced techniques namely big data analytic as shown in the following sections.

1.2 Collaborative Filtering approaches

The collaborative filtering approach is to use the preferences of a group with similar interests and common experiences to recommend information that users are interested in. Individuals give a considerable degree of response (such as a score) to the information through a cooperative mechanism and record it to achieve the purpose of filtering. To help others filter information, the response is not necessarily limited to those of particular interest, and the record of particularly uninteresting information is also important [25, 53].

The reason why collaborative filtering recommendation technology is widely used is that it has no special requirements for recommended objects and can handle unstructured and complex objects, such as music and movies. The recommendation system based on collab-

orative filtering makes corresponding recommendations from the user's perspective and is automatic, that is, the recommendations obtained by the user are implicitly obtained by the system from the purchase mode or browsing behavior, and it does not require the user to work hard to find suitable interests. Recommended information, such as filling in some survey forms. Collaborative filtering can discover users' potential interests and preferences that they have not yet discovered. These advantages make it suitable for almost all fields.

Typically, a CF system contains a list of m users $U = u_1, u_2, ..., u_m$ and n items $P = p_1, p_2, ..., p_n$. The system constructs an mn user-item matrix that contains the user ratings for items, where each entry r_i, j denotes the rating given by user u_i for item p_j . In need of a referral for the a on the target item q, the CF algorithm either predicts a rating for q or recommends a list of most likable top-N items for a. An overview of the general CF process is illustrated in Figure 1.

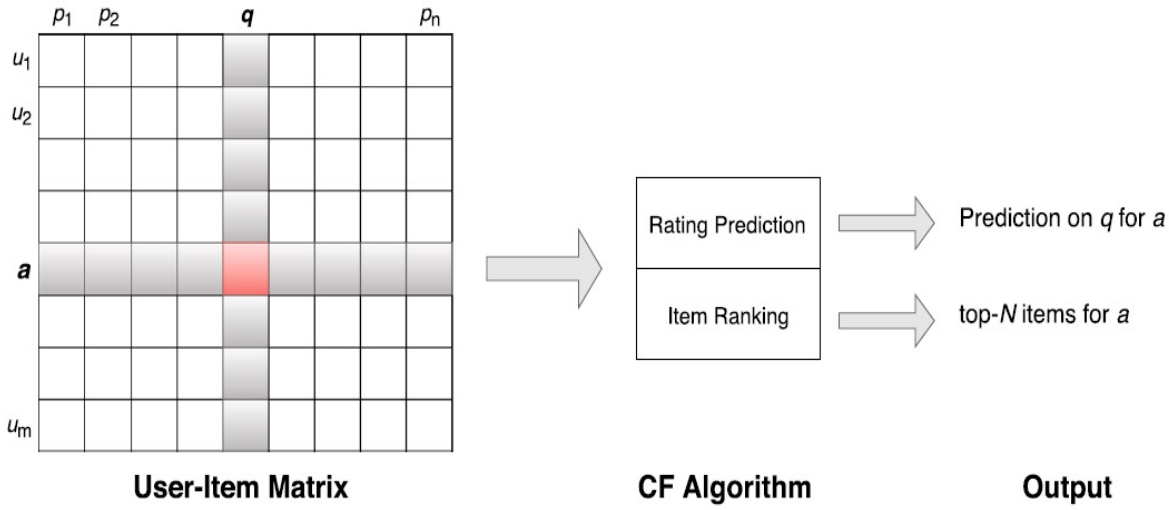


Figure 1: An overview of the Collaborative Filtering process [8].

CF algorithms follow two main methodologies approaching the recommendation generation problem:

- Memory-based algorithms utilize the entire user-item matrix to identify similar entities. After locating the nearest neighbors, past ratings of these entities are employed for recommendation purposes [10]. Memory-based algorithms can be user-based, item-based, or hybridized. While past preferences of nearest neighbors to a are employed in user-based CF, the ratings of similar items to q are used in item-based approach [8].

- Model-based algorithms aim to build an offline model by applying machine learning and data mining techniques. Building and training such model allows estimating predictions for online CF tasks. Model-based CF algorithms include Bayesian models, clustering models, decision trees, and singular value decomposition models [76] and so on.

1.3 Research Issue

Collaborative filtering approach has some major advantages over content based filtering approach in that it can perform in domains where there is not much content associated with items and where content is difficult for a computer system to analyze (such as opinions and ideal). Also, CF technique has the ability to provide serendipitous recommendations, which means that it can recommend items that are relevant to the user even without the content being in the user's profile [67, 71]. Despite the success of CF techniques, their widespread use has revealed some potential problems [8, 37, 76] such as follows.

1.3.1 Cold-start problem

The cold start problem is also known as the first-rater or new-item problem. From a certain perspective, it can be regarded as an extreme case of the sparse problem. Because the traditional CF recommendation is based on the calculation of similar users/items to get the recommendation of the target user, when a new item appears for the first time, because no user has evaluated it, the simple CF cannot predict its score and recommend. Moreover, due to the early emergence of new item, there are fewer user reviews and the accuracy of recommendations is also relatively poor. Similarly, the recommendation system for new user is also very poor. The extreme case of the cold start problem is when a CF recommendation system is first running, every user faces the cold start problem on every project [2, 5, 8, 32].

1.3.2 Data sparsity problem

In the sparsity problem, the realization of the CF algorithm first needs to use the user-item rating matrix to represent the user information. Although this is simple in theory, many e-commerce recommendation systems need to process a large amount of data information, and in these systems, the total amount of goods purchased by general users accounts for about 1% of the total amount of goods on the website, so the rating matrix (user-item matrix) is very sparse. In the case of large and sparse data, on one hand, it is difficult to find the nearest neighbor user set, and on the other hand, the cost of similarity calculation will be very high. At the same time, because the data is very sparse when the nearest neighbor user set of the target user is formed, moving forward will cause the loss of information, which leads to the reduction of the recommendation effect [36, 60, 89].

1.3.3 Scalability problem

In the CF recommendation algorithm, the global numerical algorithm can timely use the latest information to generate relatively accurate user interest prediction or recommendation for users. However, in the face of an increasing number of users, the amount of data increases sharply, and the algorithm's scalability issues (That is, the problem of adapting to the expanding scale of the system) have become an important factor restricting the implementation of the recommendation system. Although compared with model-based algorithms, global numerical algorithms save training time for building models, but the amount of calculation used to identify the nearest neighbor algorithm increases greatly with the increase of users and items for millions. The usual algorithms will encounter

serious scalability bottlenecks. If this problem is not solved well, it directly affects the recommendation system based on CF technology to provide users with a real-time solution to the recommendation problem. The better the real-time performance of the recommendation system and the higher the accuracy, the system will be accepted by users [39, 72].

1.4 Literature Review

In this section, we explore recent related works and presented them in Table 1 according to their domain.

Table 1: Summary of literature survey and an overview of collaborative filtering techniques.

Begin of Table						
CF applications	**Authors**	**Methodology**	**Data Modeling**	**Metrics**	**Datasets**	**Problems Solutions**
k Nearest Neighbors (kNN)	[25, 40, 41, 43, 49, 62, 63, 73, 74, 80–82, 84, 85, 95]	kNN executes the three tasks to generate recommendations for an active user: (1) determine k users neighbors (neighborhood) for the active user a; (2) implement an aggregation approach with the ratings for the neighborhood in items not rated by a; and (3) extract the predictions from in step 2 then select the top N recommendations.	User-item preferences.	MAE, RMSE, precision, Recall, F1 and NDCG@k.	MovieLens, Netflix and EachMovie.	Sparsity and Scalability.
Matrix Decomposition	[3, 48, 69, 75]	These algorithms projects both users and items to vectors with the same dimension, which represents user and item latent features.	User-item preferences	MAE, RMSE, precision, Recall, F1 and NDCG@k	MovieLens, Netflix and EachMovie	Sparsity and Scalability

Continuation of Table 1						
CF applications	**Authors**	**Methodology**	**Data Modeling**	**Metrics**	**Datasets**	**Problems Solutions**
Clustering	[18, 21, 30, 42, 52, 55, 59, 64, 79, 86, 88, 97]	Clustering techniques can be used to reduce the candidate set in collaborative based algorithms. Using a clustering algorithm for collaborative filtering is somewhat similar to the previous collaborative filtering based on users or items.	User-item preferences.	MAE, RMSE, precision, Recall, F1 and NDCG@k.	MovieLens, Netflix and EachMovie.	Sparsity and Scalability.
Bio clustering	[25, 40, 41, 73, 84]	These CF models computes the rating based similarity of UCF and ICF models similarity, and then employ certain clustering algorithm to generate the user or item groups.	User-item preferences.	MAE, RMSE, precision, Recall, F1 and NDCG@k.	MovieLens, Netflix and EachMovie.	Sparsity and Scalability.
Bayesian Classifier	[13, 33, 61, 66]	These methods computes the probability that a user rates an item with a specific rating value knowing the previous ratings that exist in the dataset.	User-item preferences	MAE, RMSE, precision, Recall, F1 and NDCG@k.	MovieLens, Netflix, Books Crossing and EachMovie.	Sparsity and Scalability.

Continuation of Table 1

CF applications	**Authors**	**Methodology**	**Data Modeling**	**Metrics**	**Datasets**	**Problems Solutions**
Association Rule	[50, 58]	The main problem encountered in applying association rules to the recommendation system is to convert scores into transactions 0 or 1. Usually, all positive score sets (that is, the score matrix after averaging in the past) or the user's preferences such as purchase behavior or record is regarded as a transaction.	User-item preferences.	MAE, RMSE, precision, Recall, F1 and NDCG@k.	MovieLens, Netflix and EachMovie	Sparsity and Scalability.
Hybrid algorithms	[1, 4, 7, 14, 16, 19, 30, 43, 52, 57, 60, 65, 74, 86, 95]	These models integrated the side information of users and items with user-items ratings matrix's (implicit feedback meet explicit feedback).	User-item preferences, side information.	MAE, RMSE, precision, Recall, F1 and NDCG@k.	MovieLens, Netflix and EachMovie.	Cold start problems.

End of Table

2 Objectives

The main objective of this work is to develop an effective recommendation models based on a collaborative filtering recommender system. To achieve this objective there is a list of specific research objectives as follows:

- Critical review on challenges, solutions, conventional and newly emerged recommender systems.
- To develop recommendation algorithm based on a collaborative filtering recommender system using machine learning algorithms.
- To design recommendation model based on a collaborative filtering recommender system using deep learning algorithms.
- To efficient recommendation approach based on a collaborative filtering recommender system using big data platform such as Apache spark.

3 Proposed Models

The main goal of this thesis is to alleviate some of the drawbacks of the conventional collaborative filtering approach, and it contributes to different research areas. Here, we contribute many different types of collaborative filtering systems based on memory-based, model-based, and, Apache Spark systems, with a focus on recommending movies that use big data analytics as presented in the following subsections.

3.1 Efficient Hybrid Recommendation model Based on Collaborative Filtering Recommender Systems

In this work, we address these issues in collaborative filtering (CF) recommender systems by applying a hybrid approach combining the user-based CF (UCF) algorithm and item based CF (ICF) algorithm. The bottleneck in traditional CF algorithms is the search for neighbours amongst a large user population of potential neighbors. A hybrid model avoids this bottleneck by modelling the relationships between items and the relationships between users. Because the relationships between users or items are relatively static, a hybrid of the user and item algorithms may be able to provide a higher quality compared to the UCF and ICF models with less online computation.

In this work, we present a new recommendation model named "Efficient Hybrid Recommendation Model based on Collaborative Filtering Recommender System" that blends UCF and ICF models with a linear regression model (ΓUICF). The Γ UICF model is utilized to reduce the dimension of the user-item rating matrix. The different major similarity computations between users to users and items to items are computed, and then the prediction for the unobserved target rating is made by adapting UCF and ICF models in the offline step. The Γ parameter linear regression model learns for each user and item and fuses the two predictions in the online step into a weighted model. The experiments were conducted on two different real-world datasets, namely the 100k MovieLens and 1M MovieLens datasets. The proposed model was observed to have significantly better performance when compared to the existing models.

3.1.1 System Overview

Firstly, we presented the impact of some properties of the user-item rating matrix on collaborative filtering (CF) models. Then, in subsection 3.1.1.2, we computed the similarities of both UCF and ICF models via two different metrics of similarity, namely, cosine similarity and Pearson Correlation Coefficient (PCC). After finding similarities, the prediction of both UCF and ICF models is provided in subsection 3.1.1.3. The UCF and ICF models have limitations related to sparsity and scalability. More precisely, the user-item matrix is sparse, while the user-to-user or item-to-item similarity matrix is quite dense. This is due to the fact that even a few frequently purchased items can lead to dense user-to-user or item-to-item similarities. Hence, the accuracy of the prediction each model is poor and imprecise. Therefore, to solve this issue, we proposed a new recommendation model that combines the strengths of the UCF and ICF models with a parameter linear regression model to minimize the prediction error. The proposed model is provided in detail in subsection 3.1.1.4.

3.1.1.1 Problem Formulation

The CF model aims to suggest new items or to predict the utility of a specific item for a specific user based on the user's previous interest and the behaviors of many other users.

We suppose that the user-item ratings matrix is an incomplete $m \times n$ matrix $R_{m \times n} = [r_{ui}]$ containing a set of m users $U = \{u_1, u_2, u_3, ..., u_m\}$ and set of n items $I = \{i_1, i_2, i_3, ..., i_n\}$ (see Figure 2). Each user u_i has a rated list of item I_{ui}, which the user has expressed him/her behaviors about. Behaviors can be explicit feedback given by the user as rating score within a specific integer range between 1 to 5, or can be implicitly feedback derived from purchase record such as clicks, browsing, analyzing timing logs and so. Note that $I_{u_i} \subseteq I$ and I_{u_i} can be a null-set. There exists a distinguished user $u_i \in U$ called active user for whom the task of a CF model is to find an item likeliness that can be of two forms.

1. **Prediction** of unknown ratings is a numerical value $\hat{r}_{ui}$, representing the predicted similar of item $i_j \notin I_{u_i}$ for the active user u_i. This predicted numerical value is a scale from 1 to 5 as the opinion values provided by u_i.

2. **Recommendation the Top-k items or Top-k users** is a list of k items i_j, $I_r \subset I$, that the active user will like the most. Note that the recommended list of items must be on items not already rated (purchased) by user active, i.e., $I_r \cap I_{u_i} =$. This formulation of a CF model is almost always referring to Top-K recommendation.

Note: The Top-k problem term almost frequently indicates to the process of finding the Top-k items, instead of the Top-k users. Nevertheless, Top-k user's formulation is also beneficial to the merchant to determine the best users to target with the marketing effort.

3.1.1.2 Computed Similarity of UCF and ICF Models

Generally, the similarity between the users or items based CF model is computed by treating them as vectors in the user-space or item-space respectively and computing their

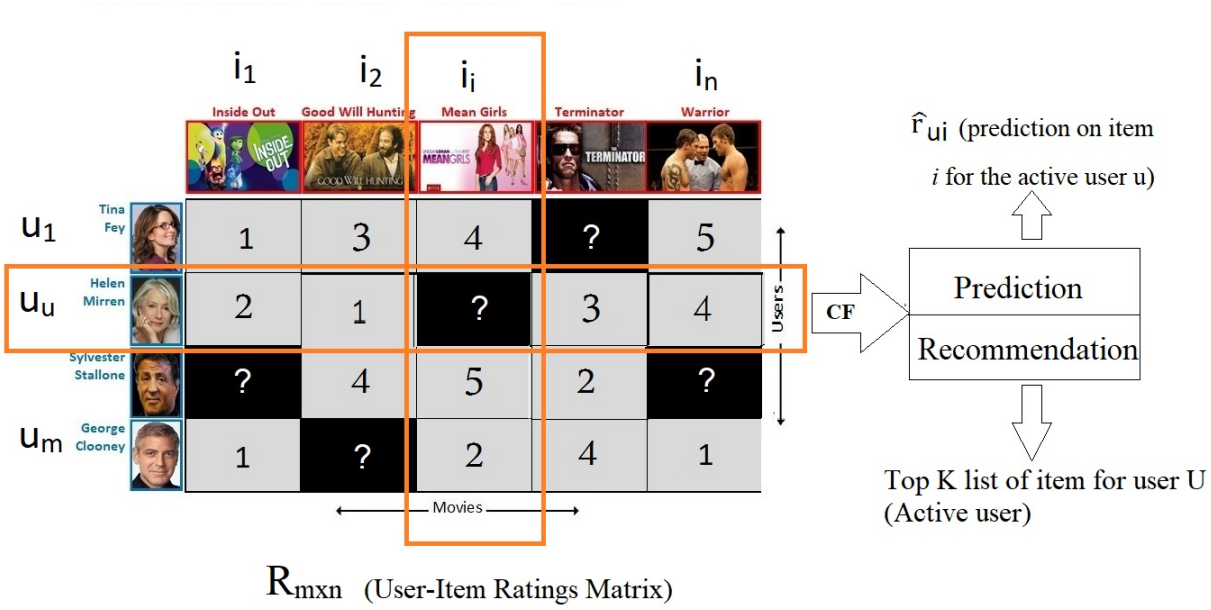


Figure 2: CF Model Process.

similarity. Various metrics are proposed such as Cosine similarity (COS) measure [70], Pearson Correlation coefficients (PCC) [51], and Jaccard [47].

In this work, COS measure and PCC are adopted, as provided in [10, 87], as we explained to them as follows:

3.1.1.2.1 Cosine Similarity (COS)

COS finds the correlation between two variables, which measure the similarity between two feature vectors in terms of the angle between them. The COS formula of the UCF model is defined in Equation 1.

$$s_{user}^{cos}(\vec{u},\vec{v}) = \frac{\vec{u}.\vec{v}}{\|\vec{u}\|^2 . \|\vec{v}\|^2} = \frac{\sum_{i\in I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{i\in I_u} r_{ui}^2 \sum r_{vi}^2}} \tag{1}$$

where the dot product $\vec{u}$ and $\vec{v}$ of the user vector is obtained and then divided by the product of the Euclidean length of the vector. I_{uv} represents the set of items rated by both users u and v, r_{ui} and r_{vi} represent the rating of user u or v for item i.

Likewise, the formulation COS of the ICF model is shown in Equation 2.

$$s_{item}^{cos}(\vec{i},\vec{j}) = \frac{\vec{i}.\vec{j}}{\left\|\vec{i}\right\|^2 . \left\|\vec{j}\right\|^2} = \frac{\sum_{u\in U_{ij}} r_{iu} r_{ju}}{\sqrt{\sum_{u\in U_i} r_{iu}^2 \sum r_{iu}^2}} \tag{2}$$

where $s(i,j)$ represents the similarity of items i and j, U_{ij} represents the set of common users who rated items i and j, across all users that rated such items.

3.1.1.2.2 Pearson Correlation Coefficient (PCC)

PCC is a measure of vector similarity. The output range is between -1 and $+1$, where zero is no correlation, negative is the negative correlation, a positive value is a positive correlation.

The UCF model formula based on PCC is given in Equation 3.

$$PCC_{user}(u,v) = \frac{\sum_{u \in I_{uv}}(r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \in I_{uv}}(r_{ui} - \overline{r_u})^2}\sqrt{\sum_{i \in I_{uv}}(r_{vi} - \overline{r_v})^2}} \tag{3}$$

where $s(u,v)$ represents the similarity of users u and v, and $\overline{r}_u$ and $\overline{r}_v$ donate the mean rating of user u or v across all items.

Likewise, formulation of the ICF model based on PCC becomes:

$$PCC_{item}(i,j) = \frac{\sum_{u \in U_{ij}}(r_{ui} - \overline{r_i})(r_{ui} - \overline{r_j})}{\sqrt{\sum_{u \in U_{ij}}(r_{ui} - \overline{r_i})^2}\sqrt{\sum_{u \in U_{ij}}(r_{ui} - \overline{r_j})^2}} \tag{4}$$

where $s(i,j)$ represents the similarity of items i and j, and $\overline{r}_i$ and $\overline{r}_j$ donate the mean rating of item i or j, across all users that rated such items.

3.1.1.3 Rating Prediction of UCF and ICF Models

After the user similarity and item similarity are computed, The next step of the recommendation method includes predicting an estimation of the unobserved rating for the items that the user's nearest neighbours have scored, but that the target user has not. Through a list of rating estimates for unrated items, the recommender model will simply sort by decreasing value and recommend the top k-items of the list to the target user. The prediction formula of the UCF and ICF models is shown in Equation 5 and 6 respectively.

$$\hat{r}_{u,i}^{User} = \mu_u + \frac{\sum_{v \in N_i(u)} sim_{uv} \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i(u)} sim_{uv}} \tag{5}$$

where $\mu_u$represents the mean rating of user u across all items rated. $N_i(u)$ represents the number of neighbors that have item i in common with user u, of which v is a particular neighbour user, and sim_{uv} is the similarity between the user u and one of its neighbors v.

$$\hat{r}_{u,i}^{Item} = \mu_i + \frac{\sum_{j \in N_u(i)} sim_{ij} \cdot (r_{ju} - \mu_j)}{\sum_{j \in N_u(i)} sim_{ij}} \tag{6}$$

Since all previous models need to be calculated for each prediction, the time cost is large, and not all the scored items belong to the category similar to the item. Therefore, the TopK technique has adopted to filter out the K items with the highest similarity to the item among all the scored items, and only weight the average of the K items (see Equation 7 and 8).

$$\hat{r}_{u,i}^{UTopk} = \mu_u + \frac{\sum_{v \in N_k(u)} sim_{uv} \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_k(u)} sim_{uv}} \tag{7}$$

$$\hat{r}_{u,i}^{ITopk} = \mu_i + \frac{\sum_{j \in N_k(i)} sim_{ij} \cdot (r_{ju} - \mu_j)}{\sum_{j \in N_k(i)} sim_{ij}} \tag{8}$$

3.1.1.4 Hybrid Proposed Model (ΓUICF)

A lot of the biggest e-commerce sites are operating on a scale that emphasises how to directly use the CF algorithm.

In memory-based CF models such as UCF or ICF similarity computation, the nearest neighbour formation process step turns out to be the performance bottleneck, which in turn can make the entire process inappropriate for generating real-time recommendations. Moreover, the user-item rating matrix is highly sparse. Hence, in some cases, the density of the matrix is lower than 1%. This is because users rated only some of the items that they viewed from a large pool of items in the user-item rating matrix. Therefore, the UCF or ICF mode needs to predict the unobserved ratings from the few available ratings. According to Sarwar et al. [70], the similarities computed using cosine or correlation measures may be imprecise or misleading in the sense that two rating vectors may be distant (in the Euclidean sense) yet may have a very high similarity. In this case, using the raw ratings of a similar item may not be a good way to go, resulting in a poor prediction.

One of the mechanisms ensuring high sparsity and scalability is to employ model-based algorithms. The Model-based algorithms have the power to contribute to RS to operate at a high sparse and scale dataset. Sarwar et al. [70] proposed a linear regression model to improve the performance of the item-based CF model. Their model used the results of the linear regression model instead of simply using the raw rating of similar items for prediction. Their model used approximated values based on a linear regression model (see Equation 9).

$$\hat{r}_N^{'} = \Gamma \hat{r}_i + \beta + \epsilon \tag{9}$$

where Γ and β indicate the parameter determined by going over both rating vectors, and ϵ indicates the error of the regression model.

For example [38], to approximate a function f for each item in the system using multiple linear regression algorithms. Hence, the overall rating r_0 can be observed to be dependent on a linear aggregation of the criteria ratings, where every criterion is assigned a parameter Γ_i, that is r_0 (see Equation 10).

$$r_0 = \Gamma_1 r_1 + \Gamma_2 r_2 + ... + \Gamma_n r_n + \epsilon \tag{10}$$

As shown in Equation (ref 111), the first criteria rating $r_1$to $r_n$for i must be estimated in the online step.Afterward, the overall rating can be computed using f. For computing r_0 to r_n, any standard CF model can be employed one-by-one for each criterion. For example, in the movie area, we are able to observe the prediction of the rating for the action aspect and specify its own CF issue, which we compute based on the ratings for this aspect of the neighbours of the target user.

In this work, we proposed a blend model-based algorithm (ΓUICF). We learn a linear regression function for UCF and ICF models and fuse the two predictions to result in a

weighted model. We used the Γ parameter (weights) as the impact coefficient for all users and items, corresponding to the UCF and ICF models, respectively. The similarity calculation model is still correlation-based, but the calculation is performed on the user and item latent space.

More precisely, in the UCF linear regression model, we try to explore for every user which aspects of an item (a movie) are of specific significance across all the items (movies) the user has scored. While in the ICF linear regression model, we try to explore how strongly the particular aspects of an item (a movie) affect the overall item (movie) ratings across all users.

Generally, we try to evaluate two kinds of dependencies in the user-item rating matrix. The main idea here is that we just need a small fraction of similar users and similar items to compute predictions that lead us to propose an alternate model-based algorithm. Although this process is effective, the user-item rating matrix can be very sparse due to users' scoring only some of the items that they have viewed from a large pool of items in the user-item rating matrix. Therefore, we can learn from the ideas of Kant S, Mahara T. [40], and Jannach D et al. [38], that both models can complement each other and possibly produce more accurate predictions as compared to the individual models. The formula of the proposed hybrid model is as follows.

$$\hat{r}_{ui}^{\Gamma UICF} = \Gamma_u * \hat{r}_{ui}^{TopkU} + \Gamma_i * \hat{r}_{ui}^{TopkI} \tag{11}$$

Similar to the idea in the TopK method, the formula can be improved to Topk a linear fusion function, and the degree of fusion is used as a weights of the prediction function (see Equation 12).

$$\hat{r}_{ui}^{\Gamma UICF} = \Gamma_u * \hat{r}_{ui}^{TopkU} + (1 - \Gamma_i) * \hat{r}_{ui}^{TopkI} \tag{12}$$

Among them, $\Gamma \in [0, 1]$ is degraded into two models at the two endpoints, and the coefficient Γ represents the degree of fusion.

where $\hat{r}_{u,i}^{TopkU}$ and $\hat{r}_{u,i}^{TopkI}$ are the prediction results of the UCF and ICF based CF model respectively, Γ_u and Γ_i are parameters regression model which control influence of the UCF and ICF models.

The proposed model (ΓUICF) utilized the Γ_u and Γ_i parameter regression model for each user and item and fuses the UCF and ICF models in online phase to minimize the error prediction and infer a user's preference for the item. The intuition behind the fusing UCF and ICF models is that both kinds of information can be valuable. In another word, the user will be interested in watching movies that are similar to movies the user previously liked and tend to avoid movies that are similar to movies the user previously disliked. This method does not require identification of the vicinity of similar users when requesting a recommendation; As a result, they tend to make recommendations much faster.

Algorithm 1 depicted a pseudo-code description of the Γ UICF model, which takes as the input of ratings $R_{m \times n}$ the user-item sparse matrix and outputs a prediction of unknown ratings based on the proposed model.

The bulk of the first loop (Line 1) computed the baseline CF model through the mean user, mean item, and all mean (Line 1). The bulk of the second loop (Line 4) takes the UCF model, where the loop consists of the following conditions: if the measured similarity

Algorithm 1 ΓUICF Algorithm

Input:$R_{m\times n}$ user-item ratings matrix (MovieLens dataset)

Output: Prediction of unknown ratings $\hat{r}_{ui}$ for $R_{m\times n}$ user-item matrix using $\overline{RMSE}$ Eq.: 14 and $\overline{MAE}$ Eq.: 13 evaluation metric

for user mean $\overline{r_{user\ u}}$, item mean $\overline{r_{item\ i}}$, all mean μ **in** $R_{m\times n}$ **do**
    Calculate $\hat{r}_{ui}^{BCF}$ baseline CF model (Eq.:17)
end for
for all users u **in** $R_{m\times n}$ **do**
    if CosSim **then**
        Calculate Sim(u,v) (Eq.:1)
        Pred $\hat{r}_{ui}^{UCF}$ baseline UCF model (Eq.: 18)
        Pred $\hat{r}_{ui}^{UCF+BCF}$ UCF+ baseline CF model (Eq.:5)
        Pred $\hat{r}_{ui}^{TopkUCF}$ Topk UCF model (Eq.:7)
    elsePCC
        Calculate PCC Sim(u,v) (Eq.:3)
        Pred $\hat{r}_{ui}^{TopkNormUCF}$ Topk Norm UCF model (Eq.:7)
    end if
end for
for all items i **in** $R_{m\times n}$ **do**
    if CosSim **then**
        Calculate $Sim(i,j)$ (Eq.:2)
        Pred $\hat{r}_{ui}^{ICF}$ Baseline ICF model (Eq.:19)
        Pred $\hat{r}_{ui}^{ICF+BCF}$ ICF+baseline CF (Eq.:6)
        Pred $\hat{r}_{ui}^{TopkICF}$ Topk ICF model (Eq.:7)
    elsePCC
        Calculate PCC (i,j) (Eq.:3)
        Pred $\hat{r}_{ui}^{TopkICFnorm}$ Topk Norm ICF model (Eq.:7)
    end if
end for
Initialization: Γ_u, Γ_i;
for each user u **do**
    for each item i has rated by user u **do**
        Pred $\hat{r}_{ui}^{\Gamma UICF} = \Gamma_u * \hat{r}_{ui}^{TopkUCF} + (1\text{-}\Gamma_i) * \hat{r}_{ui}^{TopkICF}$ (Eq.:11)
        Pred $\hat{r}_{ui}^{\Gamma UICF(Norm)} = \Gamma_u * \hat{r}_{ui}^{TopkUCFnorm} + (1\text{-}\Gamma_i) * \hat{r}_{ui}^{TopkICFnorm}$ (Eq.:11)
    end for
end for

is cosine similarity, then measure the similarity between users (Line 4) and then compute the basis of the UCF model (Line 7), compute the UCF+baseline CF model (Line 8), and compute the Topk UCF model (Line 9). Else normalizes the UCF model using the Pearson Correlation Coefficient (PCC) (Line 12).

Similarly, the bulk of the third loop (Line 14) takes the ICF model, where the loop consists of conditions. If the measured similarity is cosine similarity, then we measure the similarity between items (Line 17) and then compute the basis of the ICF model (Line 18), the ICF+baseline CF model (Line 19), and the Topk UCF model (Line 20). Else normalizes the ICF model using the Pearson Correlation Coefficient (PCC) (Line 23).

We initialized Γ_u and Γ_i (Line 26). The bulk of the loop through each user and each item has been rated by user u (Line 18), predicting the unknown ratings by weighting all users and items corresponding to the Topk UCF and TopkICF models (Line 29). Line 30, predicted the unknown ratings by weighted all users and items corresponding to the normalized TopkUCF and normalize TopkICF models.

3.1.2 Experimental Study

In this section, we present our experimentation, which includes experimental setup in subsection 3.1.2.1, dataset description is given in subsection 3.1.2.2, evaluation metrics is presented in subsection 3.1.2.3. The settings of the proposed model is given in subsection 3.1.2.4, and finally experimental results are demonstrated in subsection 3.1.2.6.

3.1.2.1 Experimental Setup

We implemented our experimentation of the models on Ubuntu 16.4 operating system running on Intel® Core™ i5-2400 CPU 3.10 GHz 4 processors and a hard disk of 500 GB. We used the python language version 3.6.

3.1.2.2 Dataset Description

Originally, we used the Netflix Prize dataset, which is a dataset about movie ratings. The standard Netflix Prize dataset contains $480,189$ users, $17,770$ items, and a total of about 100 million ratings. The dataset also includes the time of scoring, and the name and year corresponding to each movie id. For CF methods, the scale matrix generated by this dataset reaches 480189×17770, and the total items are more than 8 billion. The basic statistics on this matrix have already taken tens of seconds. Fine-grained calculations are slower. Therefore, we switched to a smaller data set MovieLens (`https://grouplens.org/datasets/movielens/`, which is the same data format. It provides data sets of different sizes, including versions of multiple sizes such as 100K, 1M, 10M, 20M (both refer to Rating numbers).

The two types of MovieLens datasets are used in our experiments. The first in MovieLens is called 100K MovieLens, there are $100,000$ ratings with 943 user and 1682 movies. Another is 1M MovieLens; it includes 6040 users and 3952 movies with $1,000,209$ ratings. In both datasets, each user has rated at least 20 movies. The sparsity of the user-item matrix is 93.701% in 100k and 95.810% in 1M. We choose these two datasets, because they are the most used datasets by researchers and developers in a CF domain.

3.1.2.3 Evaluation Metrics

In this work, mean absolute error (MAE) and Root Mean Squared Error (RMSE) are used as evaluation metrics to evaluate the prediction accuracy of estimating the ratings of specific user-item combinations. The lower the value of the MAE and RMSE, the better the prediction effect. Its definition Equation13 and Equation 14.

$$MAE = \frac{\sum_{i=1}^{N} |r_{ui} - \hat{r}_{ui}|}{N} \tag{13}$$

$$RMSE = \sqrt{\sum_{i=1}^{N} (r_{ui} - \hat{r}_{ui})^2} \tag{14}$$

where $\hat{r}_{ui}$ and r_{ui} are the predicted and actual scores of user u for item i, respectively.

Because of the cross-validation technique, the average MAE and RMSE of each dataset are selected as the final RMSE value of the prediction method.

$$\overline{MAE} = \frac{1}{k} \sum_{i=1}^{k} MAE_i \quad \text{and} \quad \overline{RMSE} = \frac{1}{k} \sum_{i=1}^{k} RMSE_i \tag{15}$$

where k is the number of cross-validation groups.

3.1.2.4 Modeling Setting

Before the experiments, we need to set the parameters for our model. Originally, all data used a set of the training set and the test set, and the scale ratio of the training set and the test set was $4:1$. That is, the scale of the training set is $80,000$ and the scale of the test set is $20,000$. The 5-fold Cross-Validation method is adopted for each dataset (see section **??** and Equation 15). The Topk approach has been adopted on the training set and test set, where the values of k scale from 5 to 200 as shown in Table 2 and Figure 3a and 3b. Furthermore, the degree of fusion is represented in Table 3 and Figure 4a 4b, where the value ofΓ is between $[0, 1]$. The optimal k value for Topk UCF is ($k = 30$), Topk ICF is ($k = 20$), Topk User normalised CF is ($k = 40$), Topk Item normalised CF is ($k = 20$), and Blend parameter linear regression is $Gamma = 0.70$ for the 100k MovieLens dataset, as shown in Tables 2, and Table 3. For the 1M MovieLens dataset, the optimal k value for Topk UCF is ($k = 30$), Topk ICF is ($k = 20$), Topk User normalized CF is ($k = 40$), Topk Item normalized CF model is ($k = 20$), and Blend parameter linear regression is $\Gamma = 0.80$.

3.1.2.5 Analysis and Discussion

In this section, we analyzed and discussed the basic models of the CF model, such as the baseline CF Model (BCF), baseline UCF and ICF model, UCF and ICF model with BCF, BCF with bias, Topk UCF and Topk ICF model, normalized UCF and ICF model, and normalized ICF model, and according to their results, we proposed our blend model.

3.1.2.5.1 Baselines CF Model (BCF)

The CF baseline model is a simple prediction based on these statistics. Its prediction formula is shown in Equation 16.

$$\hat{r}_{ui} = \mu + ub_u + ib_i \tag{16}$$

Among them, μ is the population mean, and ub_u and ib_i are the deviations of the user u and item i mean from the population mean, respectively (see Equation 17).

$$\hat{r}_{ui}^{BCF} = \overline{r_{user\ u}} + \overline{r_{item\ i}} - \mu \tag{17}$$

This model is the effect of baseline CF, and its RMSE value is **0.9694** and **0.9346** for 100k and 1M MovieLens dataset respectively

3.1.2.5.2 Baseline UCF and ICF Models

The similarity is measured by Cosine similarity metric for UCF and ICF baseline models, respectively. After the user similarity and item similarity are obtained, the similarity matrix is used for prediction. The prediction formula of UCF and ICF baseline models is shown in Equation 18 and 19 respectively. In these models, the implicit assumption that all users score out of the same statistical this distribution.

$$\hat{r}_{ui}^{UCFbaseline} = \frac{\sum_{v \in N(i)} s_{vu} \cdot r_{vi}}{\sum_{v \in N(i)} s_{vu}} \tag{18}$$

$$\hat{r}_{ui}^{ICFbaseline} = \frac{\sum_{j \in N(u)} s_{ij} \cdot r_{uj}}{\sum_{v \in N(u)} s_{ij}} \tag{19}$$

It is not possible to calculate a weighted average of all the data here, because it does not have scored items, and averaging is meaningless. Hence, it will increase the value of the denominator, leading to serious deviations in prediction. Therefore, the cross-validation result of the UCF model is **1.0149**, which is lower than the result based on the ICF model, which is **1.0174** on the 100k MovieLens dataset, which also exceeds 1. The RMSE result of the UCF model is **0.9726**, which is higher than the result based on the ICF model of **1.0025** on the 1M MovieLens dataset.

3.1.2.5.3 UCF and ICF with Baseline CF

The significance of the baseline CF model is important to use. We combined UCF with baseline CF, and ICF with baseline CF respectively, to predict the basis of each model (see section 3 Equation 5 and 6).

Hence, the RMSE error of UCF and ICF model based on the baseline CF model has been decreased to **0.9548** and **0.9362** receptively, which is a great improvement over the baseline on the 100k MovieLens dataset.

Similarly, for the 1M MovieLens dataset, the RMSE error of UCF and ICF model based on the baseline CF model has been decreased to **0.9225** and **0.8960** receptively. It can be concluded that in practical applications, indeed the ICF based on the baseline CF

model performs better, roughly the difference between commodities, not as good as the difference in taste between people.

3.1.2.5.4 Topk UCF and Topk ICF Models

The TopK technique has been adopted for UCF and ICF models based on the baseline CF model to filter out the K items with the highest similarity to the item among all the scored items, and only weight the average of the K items. If the number of items scored by the user is less than K, then all the scored items are used directly. This is also because some users may only tend to rate their favorite products. At run time, when item-item CF is used, when selecting $K = 10$, the resulting RMSE is **0.9278**, and when selecting $K = 40$, The resulting RMSE is **0.9242**. It founds that different K values have certain effects. When K is too small, it cannot cover all items that are similar to it, and when K is too large, the selected item may already be no longer very similar to it. We tried to adjust K with different K values as observed in Table 2, and Figures 3a 3b. It is found that the RMSE the training set was gradually increased, instead of gradually decreasing in the general sense.

The main reason is that the training data incorporates the mean and similarity, and the original information is inevitably used in the prediction, so the values of RMSE on the training set are not very representative. Moreover, the test set shown in Table 2 and Figures 3a 3b is the corresponding optimal K value. When K increases, its negative effect is not so great. This is probably because, with more data, the marginal data has a reduced impact on the final predicted value due to its weight reduction, similar to the law of diminishing marginal effects in economics. Obviously, different sizes of input should have different sizes of K values to match. Larger scale data should require a larger scale K value.

Here, the method we adopted is to directly set a K value set, so it is not very adaptable to different scales. In practical applications, a better way to obtain a better K value is through learning. Hence, for the 100K MovieLens dataset, the optimal values of k of UCF and ICF models are $K = 20$ and $k = 30$, and the RMSE is **0.9067**, **0.8681** for UCF and ICF models respectively. For the 1M MovieLens dataset, the optimal value of k of UCF and ICF models is $K = 20$ and $k = 40$, and RMSE is **0.9445**, **0.9213** for UCF, and ICF models, respectively.

3.1.2.5.5 Normalize UCF and ICF Models

In this work, the Pearson Correlation Coefficient(PCC) metric has been adopted to normalize UCF and ICF models. The normalized similarity matrix of users and items are calculated as shown in Equation 3 and 4, respectively.

At the same time, the function used for the prediction can be kept unchanged, and only the similarity matrix adopted therein can be replaced by the normalized similarity matrix. However, based on the TopK ICF model, the cosine similarity has been replaced by the normalized similarity matrix (PCC), but it is observed that the RMSE based on ($k = 20$) has been increased from **0.9213** to **0.9253** in 100k MovieLens dataset. While the RMSE of the ICF model based on ($k = 20$) has been decreased from **0.8686** to **0.8585** 1M MovieLens dataset.

This is due to some properties of the 100k MovieLens dataset itself. For some items

with few scores, after normalization, it is erased. These items already have very little information. For example, in a niche movie, three or four audiences with similar tastes have scored a high score of 5 points at the same time. After normalization, the high score information of the movie was erased at once, but a loss of information occurred.

For the UCF model, since each user has scored at least 20 ratings in the 100k MovieLens dataset, such disadvantages should be avoided as much as possible. Therefore, we used the normalized similarity matrix for the UCF model. However, the effect was also unsatisfactory, where the RMSE of UCF model based on ($k = 30$) has increased from **0.9445** to **0.9550**. While the RMSE value of the UCF model based on ($k = 40$) has been decreased from **0.9059** to **0.8925** in 1M MovieLens dataset.

Regarding this issue, based on several related works, we found that there are two main considerations. First, there is extensive discussion on the measurement of similarity. Among them, there is a method that still uses the similarity calculation between items, but at this time, the item is not normalized, but the user is normalized [70]. Another solution is to use the default voting method [10] when the number of scores is insufficient. In this method, an analysis method similar to tf-idf is used to obtain the default weight value. But the formula is more complicated [10]. Secondly, due to the similarity measurement method, in actual use, many essentially similar objects, their vector distance may not be ideal under the Euclidean space [70], which leads to deviations in prediction.

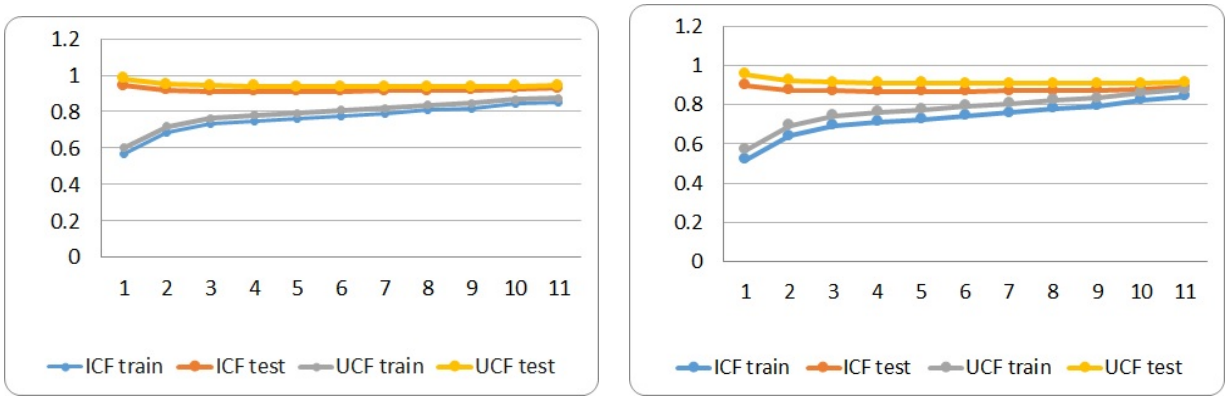
Table 2: RMSE Performance with Different K values for UCF and ICF Models on 100k and 1M MovieLens Dataset

	100k MovieLens				1M MovieLens			
k	ICF train	ICF test	UCF train	UCF test	ICF train	ICF test	UCF train	UCF test
5	0.5747	0.9543	0.6101	0.9874	0.5211	0.8975	0.5717	0.9544
10	0.6845	0.9278	0.7242	0.9573	0.6401	0.8746	0.6936	0.9244
15	0.7322	0.9223	0.7699	0.9489	0.6933	0.8696	0.7447	0.9145
18	0.7500	0.9215	0.7866	0.9468	0.7135	0.8685	0.7636	0.9118
20	0.7596	**0.9213**	0.7951	0.9458	0.7242	**0.8681**	0.7734	0.9106
25	0.7774	0.9217	0.8113	0.9449	0.7447	0.8686	0.7921	0.9085
30	0.7902	0.9225	0.8225	**0.9445**	0.7595	0.8692	0.8054	0.9076
40	0.8073	0.9242	0.8373	0.9447	0.7796	0.8712	0.8230	**0.9067**
50	0.8182	0.9258	0.8465	0.9453	0.7928	0.8733	0.8344	0.9067
100	0.8422	0.9314	0.8660	0.9491	0.8233	0.8812	0.8605	0.9093
200	0.8546	0.9345	0.8747	0.9526	0.8415	0.8892	0.8766	0.9135

3.1.2.6 Results

In the above models, there are two fundamentally different methods, item-item CF and user-user CF, although derived from the same data, they are two different dimensions. We used the results of the linear regression model instead of simply using the raw rating of similar items for prediction (See Equation 12).

Without considering the normalized similarity matrix, we merged the TopK UCF and the TopK ICF model with the best parameter linear regression model, where the TopK of UCF, ICF and Γ models based on 100k MovieLens are $k = 30$, $k = 20$, and $\Gamma = 0.70$



(a) (b)

a RMSE performance of each K value on 100K MovieLens Dataset
b RMSE performance of each K value on 1M MovieLens Dataset

Figure 3: RMSE Performance of each K value for Item to Item and User to User CF Models on 100k and 1M MovieLens Dataset.

Table 3: RMSE Performance with Different Γ Values on Train and Test set for 100k and 1M MovieLens Dataset in the Blend Model

Γ	100k MovieLens		1M MovieLens	
	TrainSet	TestSet	TrainSet	TestSet
0.00	0.8192	0.9388	0.8233	0.9048
0.10	0.8077	0.9309	0.8076	0.8948
0.30	0.7880	0.9187	0.7796	0.8787
0.50	0.7730	0.9115	0.7567	0.8681
0.60	0.7674	0.9097	0.7473	0.8649
0.65	0.7651	0.9094	0.7432	0.8639
0.70	0.7630	**0.9093**	0.7394	0.8632
0.75	0.7613	0.9096	0.7359	0.8628
0.80	0.7600	0.9101	0.7328	**0.8628**
0.90	0.7582	0.9123	0.7279	0.8639
1.00	0.7578	0.9157	0.7244	0.8665

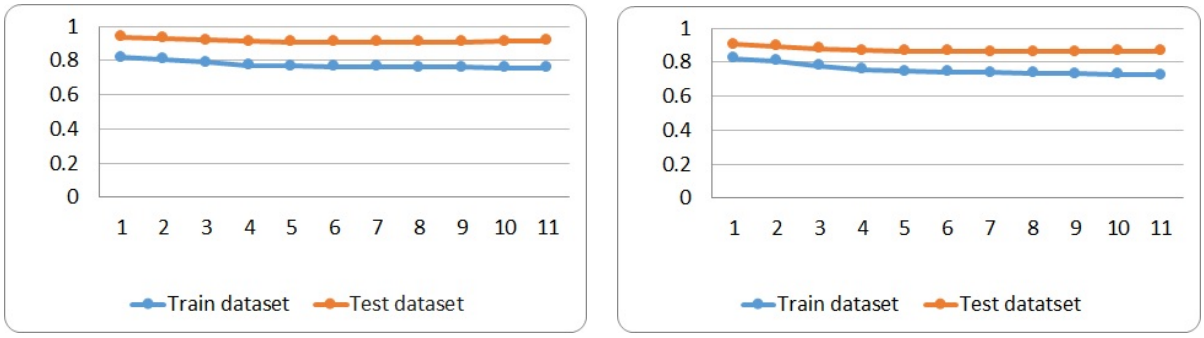
respectively. Hence, the RMSE and MAE values of the blended model ΓUICF are **0.9130** and **0.7174** respectively.

In case 1M MovieLens dataset the TopK of TopK of UCF, ICF, and Γ models are $k = 40$ and $k = 20$, with $\Gamma = 0.80$. The RMSE and MAE values after fusion are **0.8645** and **0.6751**.

With considering the normalized similarity matrix, we merged the UCF normalized and ICF normalized models with the best parameter linear regression model. For 100k MovieLens dataset, the optimal values of UCFNorm, ICFNorm, and Γ models are $k = 40$, $k = 20$, and $\Gamma = 0.70$ respectively, after fusion models, the RMSE and MAE are to **0.9229** and **0.7219** respectively.

For the 1M MovieLens dataset, RMSE and MAE of the fusion model are **0.8506** to **0.6627**.

The recommendation accuracy of the state-of-the-art models among the proposed model



(a) (b)

a RMSE Performance of Γ on 100k MovieLens Dataset
b RMSE Performance of Γ on 1M MovieLens Dataset

Figure 4: RMSE Performance with Different Γ values on Train and Test set in the Blend Model.

is compared by using RMSE and MAE evaluation metrics, where Table 4 observed the performance of our proposed model among with benchmark CF models such as baseline CF model, baseline UCF, baseline ICF, UCF+Baseline CF, ICF+Baseline CF, TopkUCF, TopkICF, NormUCF, NormICF. Our model achieves better performance under both the RMSE and MAE metrics on two datasets, which means that our blend model achieves a higher prediction accuracy and proves that the combination of UCF and ICF models with linear regression is effective.

Additionally, Table 5 observes the performance of our blend proposed model compared with existing hybrid models such as BiUCF [40], RSVD [26], ItemRank [28], iExpand [54], LDA [54], AISCF [15]. Our model achieves better performance under both the RMSE and MAE metrics on two datasets, which means that our blend model achieves a higher prediction accuracy and proves that the combination of UCF and ICF models with linear regression is effective.

Table 4: RMSE and MAE comparison of proposed model with different baseline methods on 100k and 1M MovieLens dataset

Model	100 MovieLens		Model	1M MovieLens	
	RMSE	MAE		RMSE	MAE
BCF	0.9648	0.7617	BCF	0.9346	0.7349
BICF	1.0112	0.8075	BICF	1.0025	0.7968
BUCF	1.0194	0.8109	BUCF	0.9726	0.7777
ICF+BCF	0.9352	0.7377	ICF+BCF	0.8960	0.7074
UCF+BCF	0.9499	0.7514	UCF+BCF	0.9225	0.7263
ICF+Bias	0.9350	0.7373	ICF+Bias	0.8959	0.7069
TopkICF(k=20)	0.9217	0.7215	TopkICF(k=20)	0.8686	0.6794
TopkUCF(k=30)	0.9381	0.7379	TopKUCF(k=40)	0.9059	0.7058
NorICF(k=20)	0.9259	0.7202	NorICF(k=20)	0.8585	0.6797
NorUCF(k=30)	0.9461	0.7399	NorUCF(k=40)	0.8925	0.6938
ΓUICF(Γ=0.70)	**0.9130**	**0.7174**	**ΓUICF(Γ=0.80)**	**0.8645**	**0.6751**
ΓUICF(Norm)(Γ=0.70)	**0.9229**	**0.7219**	**ΓUICF(Norm)(Γ=0.80)**	**0.8506**	**0.6627**

Table 5: According to Kant S, Mahara T. [40] results, the RMSE and MAE performance comparison of different methods with the Proposed Model

Model	100 MovieLens		1M MovieLens	
	RMSE	MAE	RMSE	MAE
BiUCF [40]	1.001	0.786	0.9378	0.7379
RSVD [26]	1.042	0.798	0.9410	0.7421
ItemRank [28]	1.006	0.796	0.9451	0.7458
iExpand [54]	1.008	0.795	0.9403	0.7414
LDA [54]	1.050	0.825	0.9423	0.7424
AISCF [15]	1.060	0.84	0.9381	0.7385
ΓUICF	**0.9130**	**0.7174**	**0.8645**	**0.6751**
ΓUICF(Norm)	**0.9229**	**0.7219**	**0.8506**	**0.6627**

3.2 Integrating of Explicit and Implicit Feedback based on deep learning collaborative filtering algorithm

In this work, a novel neural recommendation model is proposed based on Non-independent and identically distributed (Non-IID) for collaborative filtering by incorporating explicit and implicit couplings interaction. The explicit interactions consist of two models, namely Intra-coupling interactions within users and items and Inter-coupling interactions between different users and items concerning the attributes of users and items. The Intra-coupled model learns using deep learning convolutional neural networks and is combined with the Inter-coupled model. Besides explicit coupling interactions, we present a Generalized Matrix Factorization Bias (GMFB) model that systematically trains the implicit user-item coupling. Finally, it is combined with explicit and implicit coupling interactions within and between users and items accompanying the extra information of users and items under a framework called "IntegrateCF." The extensive experiments on two large real-world datasets have shown that the proposed model performs better than existing methods.

3.2.1 Methodology Overview

Firstly, we train the explicit user-item couplings concerning the user and item attributes using a deep autoencoder ConvNets model. Then we construct a generalized matrix factorization model by adding a bias for all users and items (GMFB) to train the implicit user-item couplings. More precisely, implicit user-item couplings are combined with trained explicit user-item couplings to systematically represent user, item, and user-item couplings. The deep autoencoder user-item coupling training model consists of two parts: Intra-couplings and Inter-couplings. Intra-couplings learn the explicit user-item couplings by ConvNets to capture the interactions within the user-item. Inter-couplings capture the interactions between the user-items and combine their outputs with the user-item embedding product-based abstraction. Finally, we train two neural network models, the Intra/inter Couplings model and the GMFB model, to embed both models and relationships into a complete representation of user-item interactivity under a framework called IntegrateCF. We perform extensive experiments on two real-world datasets to demonstrate the effectiveness of different IntegrateCF models such as Intra-couplings, Inter-couplings, GMFB, and their combination IntegrateCF are performed. The experimental results

demonstrate that all IntegrateCF models outperform baseline models and existing work on MovieLens 1M and Trading data.

In the following subsections, we begin by defining the problem of the collaborative filtering approach based on explicit and implicit feedback in subsection 3.2.1.1. Coupling formulation is present in subsection 3.3. We provide the explicit Intra&Inter-couplings models in subsection 3.3.0.1. The implicit feedback based on the GMFB model is presented in subsection 3.3.0.2. Finally, subsection 3.3.0.3 describes the proposed hybrid model.

3.2.1.1 Formulating the Problem

Considering a recommendation issue, assume that there is a set of m users $U = \{u_1, u_2, u_3, ..., u_m\}$ and set of n items $I = \{i_1, i_2, i_3, ..., i_n\}$, and the rating preferences represent the interactions between users and items, resulting in the formation of the rating matrix R. Each element r_{ui} in R represents the interactions among the user u and item i. The rating preferences can be explicit or implicit feedback. The user gives explicit feedback as a rating score within a specific integer range between 1 and 5 (i.e., such as 5-star ratings) (see Figure 5 A). In contrast, implicit feedback is derived from purchase records such as clicks, browsing, analyzing timing logs, and so on. The user identifies it as rating score binaries between 1 and 0 (see Figure 5 B). Besides the explicit and implicit rating matrix mentioned above, we include user demographic features and item description features in the interaction prediction. The task in this work is to focus on implicit user feedback on items, represented as a logistic regression problem, to predict whether users and items can interact. The user-item implicit feedback matrix $R \in R^{m \times n}$ as,

$$r_{ui} = \begin{cases} 1, & \text{if interaction (user u, item i) is observed;} \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

The entry r_{ui} refer user's i^{th} rating score of the item i^{th}. If the value of r_{ui} is 1, then it shows an interaction between user and item. Otherwise, there is no interaction between the user and the item. Since these interactions do not specify whether the user likes or dislike the item, there is a possibility of noise signals. Moreover, let the user's latent feature matrix be $x \in R^{m \times a}$, the vector x_u of the i^{th} represents the features of i^{th} users and n represents the dimension of features. Similarly, the latent feature of the item is represented as $y \in R^{n \times b}$ [31].

We formulate predicted recommendations from implicit feedback as predicting the scores of unobserved entries in R, which are used for ranking the movies.

3.3 Coupling Formulation

To explain the coupling relationships in this work, we provide an example in Figure 6. There is a rating matrix involving three users and three movies with their attributes. The majority of current CF models employ a rating matrix for the recommendation but ignore user and item attributes. However, when the rating matrix is extremely sparse, item attributes and users may be helpful in solving the problem.

As shown in Figure 6, there are four tables, namely Table M represents the user's rating of an item, Table Z indicates the demographic attributes of users, Table Y identifies the item's properties, and Table X depicts the implicit user-item interaction. In the explicit

Figure 5: A simple example observed the different between implicit feedback and explicit feedback rating matrices. (A) Rating matrix for explicit feedback and (b) Rating matrix for explicit feedback.

user-item coupling scenario, a user assigns a rating to the item, which reflects the user's subjective preferences for the item [12]. In the general and intuitive assumption, there is actual coupling existing in Table M called,user-user/item-item rating coupling. As a result, the rating couplings $M(\cdot)$ in Table M depict the relationship between a user's ratings of an item, which can be partitioned into intra-rating couplings between users ($M_a(\cdot)$ and inter-rating couplings between items ($M_e(\cdot)$, and the aggregate rating couplings is $M(M_a(\cdot), M_e(\cdot))$, and each rating representing the subjective behaviors of a user u_i on an item ij. By fusing the coupling between users and items, the overall explicit coupled user-item similarity, RS^M can be computed by Equation 21.

$$RS^M = M(M_a(\cdot), M(M_e(\cdot)) = M(M_{j_1,j_2}, M_{i_1,i_2}) \tag{21}$$

whereas RS^M indicates the coupling between rating r_{ij}, M_{j_1,j_2} indicates the rating coupling all over j_1, j_2 items known Intra-coupling, and M_{i_1,i_2} indicates the rating coupling all over i_1, i_2 users known inter-coupling. Even though the ratings in Table M are subjective, the overall rating behavior may be well supported by underlying reasons. Such underlying important factors may be explored from both objective and implicit perspectives.

For instance, any coupling function for a particular cell such as i_1, j_1 in Table X is the product of matrix Y_a for particular movie property p_{j_1} and matrix Z_a for particular user property p_{1_1}. Hence, this is called a coupled user-movie cell. As shown in Table X, the coupling in each cell called X_{i_1,j_1}, can be a matrix to learn. M coupled user-movie cell x_{ij} corresponds to all users and movies which satisfy a particular user i's property and an movie j's property. For instance, in Table X, a cell CP captures all users and movies with the coupling between the zip code property in user Table Z and the genre in Table Y for all users and movies.

Therefore, the implicit coupling in a user-movie coupled cell X_{i_1,j_1} is called RS, which include two parts namely user's coupling and movie's coupling. The user's coupling i_1's particular property on all movies with item property number j_1, namely $X_a(X_{i_1 j_{1*}})(1 \leq j_{1*} \leq I)$; while the movie's coupling j_1's particular property on all users with the user

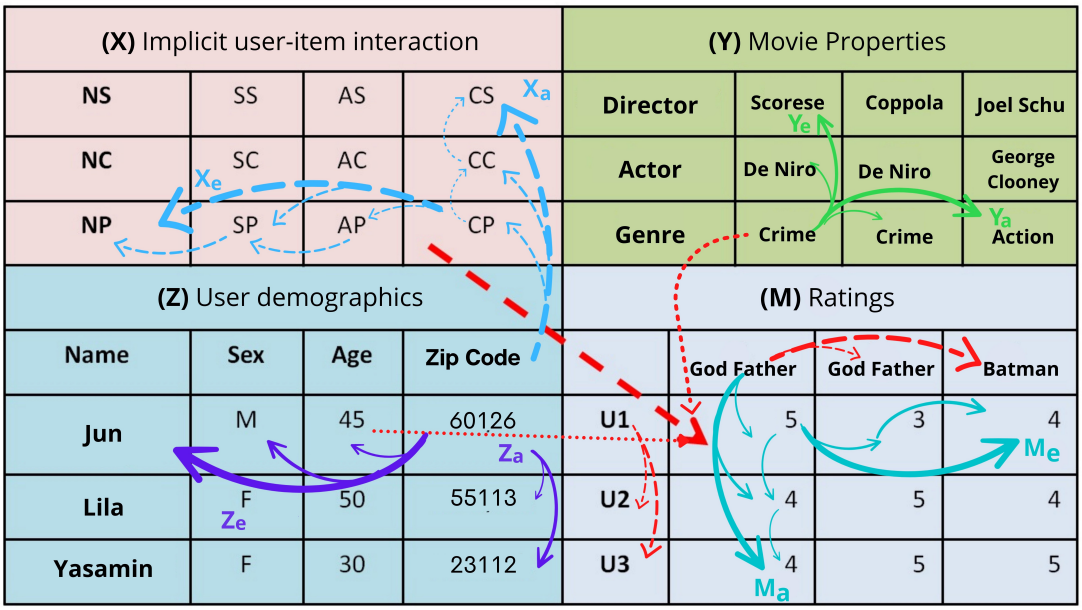


Figure 6: Example demonstrate the coupling interaction of explicit and implicit feedback along with the information of user and movie.

property number i_1, namely $X_e(X_{i_1 j_{1*}})(1 \leq i_{1*} \leq I)$. As a consequence, the implicit coupling in a user-movie coupled cell is $RS^{X_{i_1,j_1}}$ as shown in Equation 22.

$$RS_{cell}^{X_{i_1 j_1}} = RS^{X_{i_1 j_1}}(X_a(X_{i_1 j_1})), (X_e(X_{i_1 j_1})) \tag{22}$$

The aggregate implicit user-movie coupling hidden in Table X has the similarity RS^X, which is the aggregation of all user-movie cell coupling in Table X which can be expressed in Equation 23.

$$RS^X = RS^X(RS_{cell}^{X_{i_1 j_1}}) \tag{23}$$

Here $X_a()$ denotes the coupling between different movie properties on a specific user property, and $X_e()$ denotes the coupling between user properties in terms of a specific movie property.

In Table X, an example is cell CP, where the full coupling is composed of $CP_a()$, which captures all coupling between all movie genre in a user's City, and $CP_e()$, which calculates the coupling of all user zip codes on movie property genre.

Thus, the aggregated user-movie coupling is the integration via an aggregation function $RS^{M+X()}$ of explicit user-movie couplings RS^M and implicit user-movie couplings RS^X is represented in the following Equation:

$$RS^{aggregate} = RS(RS^M, RS^Z, RS^Y, RS^X) \tag{24}$$

whereas the RS^Z indicates the cumulative indication of the influence between users identified by user properties and its effect on prediction and, RS^Y indicates the relationship between movies shown by movie properties.

3.3.0.1 Learning Explicit User-item Coupling

In this work, explicit feedback is represented by two forms of coupled interaction: explicit Intra-couplings, interactions within users and items, and Inter-couplings interactions between users and items (see Figure 7).

Initially, we have embedded the user and item attributes into two input dense vectors, $p_c = \{p_{c1}, p_{c2}, ..., p_{cm}\}$ and $q_c = \{q_c, q_c, ..., q_{cm}\}$ respectively in the same vector space. p_u, q_i represents the i^{th} element in the vector $q_c(p_c)$. Practically, the information of users represents demographics of users, social relationships, or user review information (see Table Z in Figure 6). The information about items represents the item descriptions (see Table Y in Figure 6).

To compute the coupling relationships between embedding vectors of p_u and q_i, assume that $M_\vartheta(p_u, q_i)$ computes the interaction between elements of embedding vectors p_u and q_i. Here, ϑ represents the parameters of the function M, which can be learned or fixed practically. And then we get a user-item interaction coupling matrix, that is denoted by M_c, in which each element is represented as $M_{u,i}$, to represent the coupling interactions between the element p_u of vector p_c and the element q_i of vector q_c (see Equation 25).

$$M_c = (p_u \times q_i) \tag{25}$$

Here we have two forms to learn the coupling matrix M_c, which are Intra-couplings (M_a) and Inter-couplings (M_e). For Intra-couplings, it is fed to the flattening layers followed by the deep autoencoder ConvNet layers, which include encoder and decoder layers functions to map the latent vectors (see Equation 26).

$$\begin{aligned} &\left.\begin{aligned} M_a &= Conv2D(W * M_c + b) \\ M_a &= MaxPooling2D(M_a) \end{aligned}\right\} (Encoder) \\ &\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ &\left.\begin{aligned} M_a &= Conv2D(W * M_a + b) \\ M_a &= UpSampling2D(M_a) \end{aligned}\right\} (Decoder) \end{aligned} \tag{26}$$

where Conv2D represents the ReLU as the non-linear activation function, W represents filters, b indicates a bias for W, and $*$ indicates the convolution operator. We concatenated the Inter-couplings user-item interaction with the output of Intra-couplings layers followed by the fully-connected layers to predict the scores (see Equation 27 and 28).

$$z_{ui} = concatenate(M_a, M_e) \tag{27}$$

$$\hat{z}_{ui}^{Explicit} = \delta(w_L^T z_{ui} + b_L) \tag{28}$$

3.3.0.2 Learning Implicit User-item coupling

In this model, we learn the implicit feedback by using the generalization of the Matrix Factorization Biases (GMFB) model. The Generalisation Matrix Factorization (GMF)

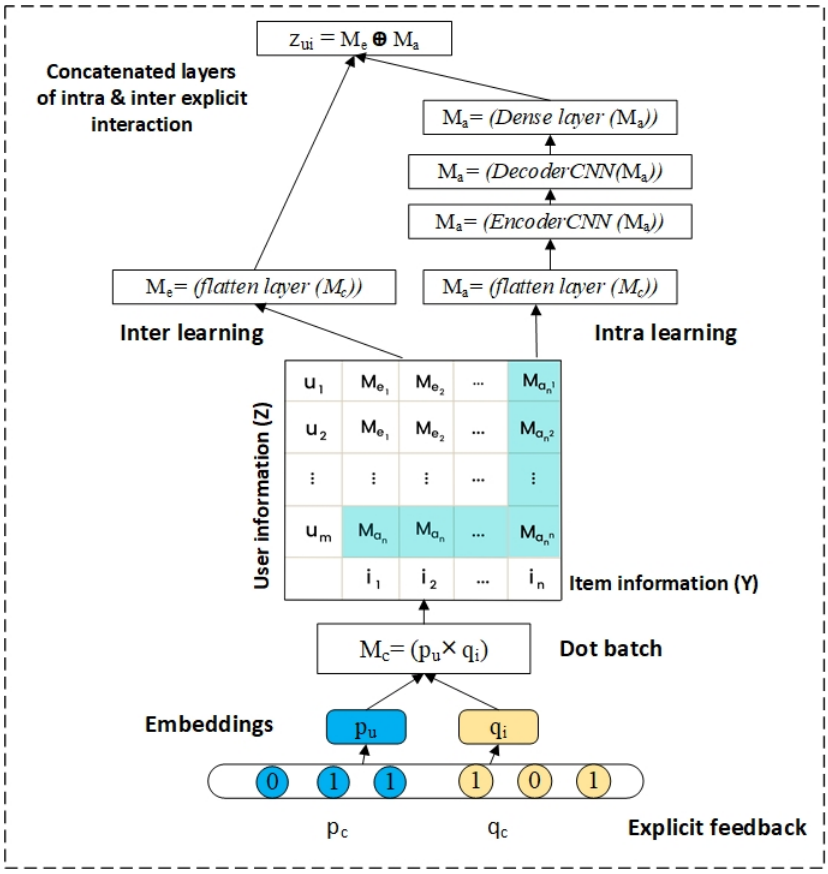


Figure 7: Learning Explicit feedback.

[31] have only one layer of a neural network. This layer of neural network multiplies the one-hot encoding of users and items element-wise and then takes the product matrix as input, and its output is the final score. In formal terms, this model's input layer consists of a user-item pair u and i, that represent binaries one-hot encoded feature vectors userID n_u and itemID n_i, respectively, where $n_u \in R^{|U|\times 1}$ and $n_i \in R^{|I|\times 1}$. Applying an embedding layer on n_u gives latent factors for user u, which is identified as x_u. Similarly, applying the embedding layer on n_i gives the latent factor for item i, which is identified as y_i. And create the functions of user and item interaction $\Theta(x_u, y_i)$ (see Equation 29).

$$r_{ui} = \Theta(x_u, y_i) = (x_u \odot y_i) \tag{29}$$

In this regard $\odot$ identifies the element-wise product of x_u and y_i vectors.

In the stander matrix factorization approaches, the general and intuitive assumption is that the MF approaches for implicit feedback tend to work better if user and item biases are computed. Therefore, we introduce parameters γ_u and γ_i for GMF model, and try to fit a model (see Equation 30).

$$\hat{r}_{ui}^{Implicit} = \Theta(r_{ui} + \gamma_u + \gamma_i)) \tag{30}$$

where, γ_u and γ_i represent the user bias of user u and item i respectively. A high item bias γ_u raises all users' preferences $\hat{r}_{ui}$, indicating that the item is more widely loved (or mainstream); on the other hand, a low item bias may suggest a niche item. A high bias γ_i

raises the values of all preferences $\hat{r}_{ui}$ for all items. In other words, a user with a high bias loves a wide range of goods, whereas a person with a low bias loves only a small number of items.

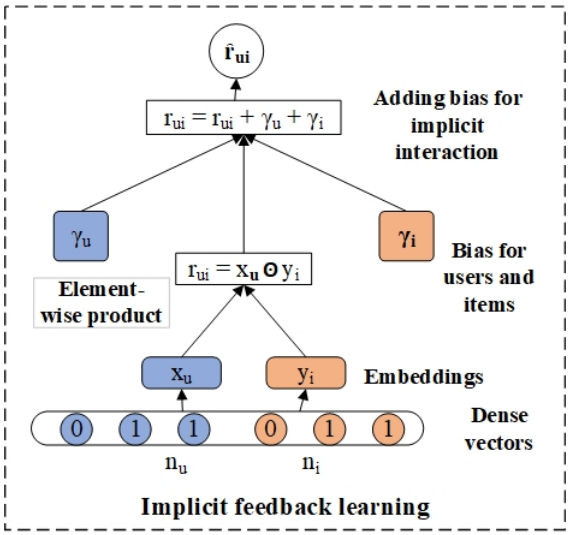


Figure 8: Generalized matrix factorization with a bias for learning implicit feedback.

3.3.0.3 Fusion of Explicit and Implicit Feedback Couplings Models

In this section, the Intra and Inter-coupling explicit interaction model meets the implicit user-item interaction model GMFB in the integrateCF framework to enhance the model of complex user-item interactions and effectively improve recommendation accuracy. The Intra and Inter-coupling explicit (see Equation 28) and implicit GMFB (see Equation 30) models have embedding layers that obtain a sort of flexibility to blend the models [31]. Then we have concatenated the last hidden layers of Intra and Inter-coupling explicit and implicit GMFB to get the enhanced result and predict the i_{th} user's rating score on the i_{th} item as shown in Equation 31 and Figure 9. First, we used the explicit features to obtain the user-item matrix M_C of the cross-product of embedding p_u and q_i as shown in Equation 25. Then, we regarded the user-item matrix M_C as Intra and Inter-coupling explicit interactions. The autoencoder learns the Intra-coupling interactions in the ConvNet model. The Inter-coupling interactions are flattened as a vector and concatenated with the last decoder layer, followed by a fully connected layer to learn a highly abstract representation. Then trained the Intra and Inter-coupled explicit models with the implicit GMFB model to learn separate embedding, and finally, we joined the two models by concatenating their last hidden layer followed by an MLP network to generate the final user-item interaction vector. Figure 9 depicts the proposed model's blueprint framework, which is mathematically expressed in Equation 31. The model obtained the probability values representing the user's scores in the output layer. Finally, the score with the highest probability is used as the proposed model's prediction result.

$$\hat{r}_{ui} = z_{ui}^{Explicit} \oplus r_{ui}^{Implicit} \tag{31}$$

In this case $\oplus$ represents the vector concatenation operation, $z_{ui}^{Explicit}$ represents the latent vector of the explicit interaction (see Equation 28), and $r_{ui}^{Implicit}$ represents the latent vector of implicit interaction (see Equation 30).

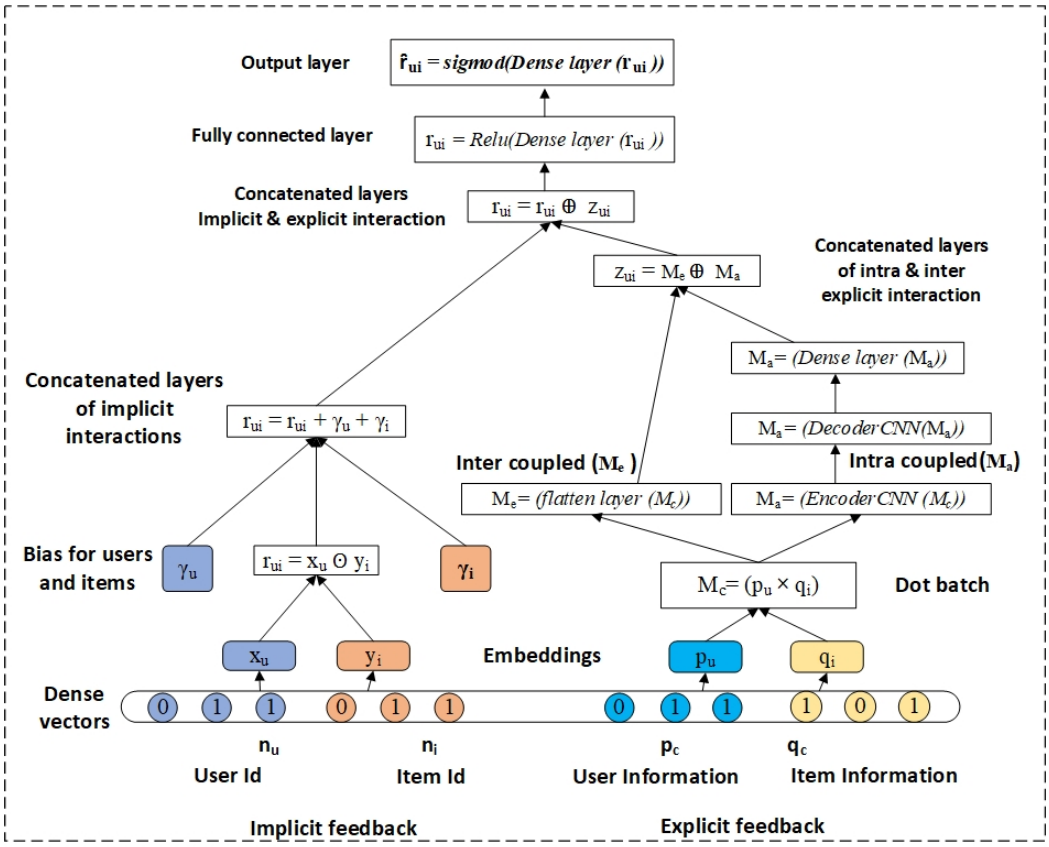


Figure 9: The architecture of the IntegrateCF framework

As discussed above, a new paradigm is proposed in recommendation design that shifts the focus from understanding explicit user-item preference to considering both explicit and implicit user-item interactions across multiple sources, with different properties, layers, and forms.

3.3.0.3.1 Model Training

The output of the proposed model is a nonlinear function activated through Sigmoid and squeezed between 0 and 1. Thus, the training dataset of the proposed model can be identified as $D = \{< (p_u^i, q_i^i, x_u^i, y_i^i), r^i\}\, i \in \{1, 2, ..., |D|\}$, where i represents the i^{th} sample training dataset. The interprets an Intra and Inter-coupling explicit interaction with implicit GMFB prediction concerning a probability as shown in Equation 32.

$$p_\sigma = (r = 1 | p, q, x, y) \tag{32}$$

Here, σ is the neural network weights. We used $\hat{r}$ as the prediction output as shown in Equation 31.

The proposed model's cost function is represented by Equation 33.

$$\partial(\hat{r}^{(i)}, r^{(i)}) = -r^{(i)} log(\hat{r}^{(i)}) - (1 - r^{(i)}) log(1 - \hat{r}^{(i)}) \tag{33}$$

At last, we trained the proposed model parameters ∂ perusing the cost function shown in Equation 34.

$$Loss = \frac{1}{m} \sum_{i=1}^{m} \partial(\hat{r}^{(i)}, r^{(i)}) \tag{34}$$

3.3.1 Experimental Study

We implemented our experimentation of the models on the windows 10 operating system running on Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz processors and a hard disk of 1 TB. We used the python language version 3.7. As provided our experimentation, included a description of the datasets in subsection 3.3.1.1, evaluation protocols in subsection 3.3.1.2. Subsection 3.3.1.3 presents the settings of the proposed model, the time complexity is given in section 3.3.1.4, subsection 3.3.1.5 shows the comparison with baselines, and finally, the results and discussion are provided in subsection 3.3.1.6.

3.3.1.1 Datasets Description

In this work, two different real-world datasets have been used: MovieLens 1M and Tafeng. MovieLens 1M is made up of one million transactions involving 6,040 users and 3,952 items. The MovieLens 1M incorporates user demographics such as gender, age, occupation, zip code, and movie features such as genres and titles. In contrast, the Tafeng has 817,741 transactions, 32,266 users, and 23,812 items. The Tafeng has demographic attributes such as Customer ID, Age, and Region and some item descriptions such as Original ID, Subclass, Amount, Asset, and Price.

3.3.1.2 Evaluation Protocols

In convolution machine learning algorithms for the recommender system, as evaluated proposed models using metrics like accuracy or F1 for classification problems and RMSE or MAE for regression problems. Nevertheless, these metrics are overly simplistic for evaluating RSs. In this work, we applied the leave-one-out evaluation to test the performance of item recommendation, which has been widely used in comparison models for implicit feedback recommendation [31, 92].

The main idea here is that the user does not need to react to each item in the list of recommendations. Instead, we simply need the user to react with at least one item on the list; as long as the user does so, the recommendations are successful. As a result, there is a set of evaluation protocols that must be followed to produce a list of 10 recommended items for each user and they are listed as follows:

1. First, select 99 items randomly for every user who has not yet reacted to them.
2. Add these 99 items to the test item (the actual item that the user reacted with). Hence, this brings the total number to 100 items.
3. Apply the proposed model to these 100 items and rank them based on their predicted probabilities.
4. Choose from the list of 100 items the top 10 items. If evaluate 5- appears among the top 10 items, called this a hit.

5. Repeat the procedure for each user. The average hits are then calculated using Normalized Discounted Cumulative Gain (NDCG) and Hit Ratio@10 (see Equations 35 and 37).

$$NDCG@k = \frac{DCG}{iDCG} \tag{35}$$

where the discounted cumulative gain (DCG) is that having users' favorite items at the top of the recommended list will improve the user experience more than having them at the bottom, which is defined as:

$$DCG(b, L) = \sum_{i=1}^{b} r_i + \sum_{i=b+1}^{L} \frac{r_i}{log_b i} \tag{36}$$

In Equation 36, r_i indicates whether the item in the i_{th} position is favored by the user; $r_i = 1$ means the user favors the item; $r_i = 0$ means the user disfavored the item; b is the free parameter generally set to 2; L is the length of the recommended list.

$$HR = \frac{hits}{users} \tag{37}$$

where users represent the total number of users and hits represent the number of users whose evaluated set items appear in the Top-N recommendation list.

3.3.1.3 Parameter Setting

We used a normal distribution with a mean of 0 and a standard deviation of 0.01 to randomly initialize the proposed model parameters. As determined, the embedding vector size (8, 16, 32, 64, 128) and best size for user and item were set to 32, respectively. For MovieLens and Tafeng datasets, Adam optimizer [44] used with 0.001 and 0.005 learning rate and parameter values, respectively. The batch size has been set to 256 and 30 epochs. There is a cross-product of the features of users and items as Intra and Inter-coupling explicit interactions. For Intra-coupling explicit interaction, it is utilized by the autoencoder ConvNet which consists of an encoder and a decoder hidden layer ($8 \rightarrow 4 \rightarrow 4 \rightarrow 8$). The encoder was constructed as ($8 \rightarrow 4$) with a kernel size of 2 with the same padding. With a size of 2, used as the max-pooling function. For the decoder hidden layer, construct it as ($4 \rightarrow 8$) with a kernel size of 2 and the same padding. Therefore, it used the UpSampling2D function with a size of 2. In contrast, in the explicit inter-coupling interaction, we have flattened and combined it with the last decoder hidden layer, followed by a fully connected layer with 16 nodes. We added to the GMF model the embedding bias, γ_u and γ_i of users and items, with the embedding vector size set to 32 and the input and output dimension set to 1. These embeddings greatly improved the performance of the model in the final result.

3.3.1.4 Time Complexity

The time required to run our model on the specified configuration is calculated as follows: by calculating the required time for a single epoch execution and the number of epochs, we

found that it takes approximately 209–216 seconds to run each epoch. For example, iteration 6 takes 215s, iteration 9 takes 212.6s, and iteration 5 takes 216s. For the MovieLens dataset, we considered 30 epochs. Therefore, the total time complexity of the proposed model is 11239.1s. For Tafneg datasets, it takes approximately 1083.1s to 1088.0s to run each epoch. For example, iteration 24 takes 1088.0s, iteration 22 takes 1085.7s, and iteration 1 takes 1083.4s. We have considered 30 epochs, so the total time complexity of the model is 39188.8s. As per big O notations concerning the number of loops and iterative approach, the complexity was O(n). As we can notice from the above-mentioned values, the execution time of the proposed model takes more time because of the low hardware architecture.

3.3.1.5 Comparison with Baselines.

In this subsection, we provide experimental results of the proposed model and corresponding comparison with the baseline model, which is reported in terms of HR@10 and NDCG@10 for Top-K item recommendation, where k is scaled from 1 to 10. The IntegrateCF model has been configured in five different ways to train different kinds of interaction couplings in user-item in subsection 3.3.1.5.1. In addition, state-of-the-art methods are provided against our methods in subsection 3.3.1.5.2.

3.3.1.5.1 Our Models

- **GMFB**. The GMFB model and two embedding biases for users and items train the implicit feedback information. The embeddings bias layers utilized due to the matrix factorization methods based on implicit feedback tend to work better if user and item biases are computed. And utilized to prevent the overfitting caused by deep learning layers.
- **Intra-coupled with bias**. The Intra-couplings integrate the ConvNet local user-item coupling learning and fully connected layers with adding two embedding biases for users and items.
- **Inter-coupled with bias**. The Inter-couplings learning component without the CNN component.
- **Explicit Intra&Inter.** The Explicit Intra/Inter is the whole part of explicit coupling interaction, which consists of the Intra and Inter coupling interaction.
- **IntegrateCF**. The IntegrateCF model, which combines the Intra and Inter-coupled vectors with a GMFB vector.

3.3.1.5.2 Baseline Models

Furthermore, the following pertinent and representative existing methods are utilized as the benchmarks against which our methods are evaluated:

- **CML.** The collaborative metric learning (CML) model is presented by utilizing the power of metric learning to CF with implicit feedback. The CML method can learn a joint metric space to encode not only users' preferences but also the user-user and item-item similarity and improve recommendation performance [35].

- **Match4Rec.** [91] Match4Rec is a recommended method based on Bidirectional encoder representation with the Matching task.
- **KGECF.** A Knowledge Graph Embedding based CF (KGECF) method [94] is based on the RotatE model which projects a relation as a rotation in complex vector space to ensure the capture of related properties such as symmetry, asymmetry, inversion, and composition.
- **LRML.** Latent relational metric learning (LRML) [78] is a method which learns the latent relations to describe each user-item interaction instead of push-pull mechanisms. This helps to avoid the potential geometric inflexibility of translation-based knowledge graph embedding.
- **NeuMF**. The NeuMF model [31] contains a general framework for learning the functional relationships projected by matrix factorization in collaborative filtering with the neural network.
- **Wide&Deep.** Google's wide&deep neural network model [17] combines memory and generalization for the recommendation using cross-product features of the wide network's input.
- **DeepFM.** The DeepFM model [29] is a new neural network that combines the power of factorization machines for recommendation and deep learning for feature learning.
- **Lcoupled.** The Lcoupled [92] is the local user-item coupling learn using the ConvNet and fully connected layers.
- **Gcoupled**. The Gcoupled [92] is included the global couplings learning component without the CNN component.
- **CoupledCF**. The CoupledCF [92] combines ConvNet local user-item interaction coupling and deep learning neural network.

3.3.1.6 Results and Discussion

As previously stated, IntegrateCF is customized to five versions: GMFB, Intra-coupling with bias, Inter-coupling with bias, explicit Intra&inter coupling, and the comprehensive IntegrateCF. As shown in Table 6 and Figure 10, IntegrateCF generally beats other versions for top-K item recommendations on MovieLens and Tafeng datasets. By comparison, Explicit Intra&Inter beats GMFB, Intra-coupling with bias, and Inter-coupling with bias in all cases on the MovieLens dataset, Inter coupled with bias beats GMFB, Intra-coupling with bias, and explicit Intra&Inter in all cases on the Tafeng datasets.

Further, the proposed model is compared with different existing methods as shown in Table 7. The results show that the proposed model significantly improves recommendation performance on MovieLens 1M and Tafeng. We have compared our results with lCoupledCF, gCoupledCF, and CoupledCF results. The comparison is demonstrated in Table 7 which shows that our model (IntegrateCF) is performing better than state-of-the-art approaches on MovieLens 1M and Tafeng datasets. Besides, those methods do not fully account for the attributes of explicit and implicit feedback, so it's difficult to explore their

effects. The study used only user demographic attributes and a subset of item qualities, whereas real-world data can collect all user-item features.

Moreover, we have compared our results with DeepFM, NeuMF, and Google's wide&deep model's results. The comparison is demonstrated in Table 7, which shows that our model (IntegrateCF) is performing better than state-of-the-art approaches on the MovieLens 1M and Tafeng datasets.

Generally, those methods that handle the user-item information as an IID cannot fully utilize user-item information to improve the performance of recommendations. Moreover, most of the existing models assume that the attributes are independent. This supposition does not often hold, and there exist complex coupling relations between instances and attributes. The same limitation is observed with the KGECF, LRML, CML, and DeepCF models, as shown in Table 8. The results show that the proposed model beats these models on MovieLens 1M. We have compared our results with the Match4Rec model [91], as shown in Table 8 that our model is performing better than the Match4Rec model on MovieLens 1M. Moreover, because users' dynamic preferences change over time, it is difficult to record user patterns in historical sequences. As a result, the CF model's prediction quality is poor. In this work, we try to evaluate two kinds of dependencies in the user-item rating matrix. In addition, the information about the user and the items that contribute also supports this exploration. The intuition behind fusing explicit Intra and Inter-coupling interaction with implicit feedback interaction is that the idea of NetflixPrize's final award-winning algorithm merged with over three hundred models at multiple levels and scales, and thus we can learn that both models can complement one another and possibly produce more accurate predictions than the individual model. The proposed model has very high accuracy in recommendations. However, there is a slowdown in the recommendation process because the information is collected from independent features for users and items when the recommendation is requested.

Table 6: HR@10 and NDCG@10 comparison of the proposed model with different baseline methods on 1M MovieLens and Tangef dataset

Recommendation Models	MovieLens		Tafeng	
	HR@10	NDCG@10	HR@10	NDCG@10
GMFB	0.7233	0.4416	0.6396	0.4228
Intra coupled with Basie	0.7985	0.4990	0.6258	0.3859
Inter coupled with Basie	0.7404	0.4539	0.7076	0.4759
Explicit Intra&Inter	0.8877	0.6622	0.6578	0.4255
IntegrateCF	**0.9079**	**0.7013**	**0.7057**	**0.4744**

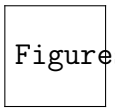


Figure 10: The performance of the proposed model along with baseline methods for MovieLens and Tangef datasets using HR@10 and NDCG@10 metrics

Table 7: According to [92, 93] results, here is the HR@10 and NDCG@10 performance comparison of the proposed model with different methods on 1M MovieLens and Tangef dataset

Recommendation Models	MovieLens		Tafeng	
	HR@10	NDCG@10	HR@10	NDCG@10
DeepCF	0.7147	0.4312	0.6506	0.4322
lcoupled	0.8212	0.5408	0.6798	0.47
gcoupled	0.7826	0.5252	0.6643	0.4205
CoupledCF	0.8252	0.544	0.6953	0.4623
NeuMF	0.731	0.448	0.6409	0.4213
Wide and deep	0.73	0.447	0.642	0.4233
DeepFM	0.6452	0.3703	0.6821	0.4522
Explicit Intra&Inter	**0.8877**	**0.6622**	0.6578	0.4255
IntegrateCF	**0.9079**	**0.7013**	**0.7057**	**0.4744**

Table 8: HR@10 and NDCG@10 performance comparison of the proposed model with another different existing methods on MovieLens 1M.

	MovieLens 1M dataset	
Recommendation Models	HR@10	NDCG@10
DeepCF [22]	0.7253	0.4416
KGECF [94]	0.7457	0.4535
Match4Rec [91]	0.8096	0.4973
LRML [94]	0.7281	0.4303
CML [94]	0.7344	0.4421
IntegrateCF	**0.9079**	**0.7013**

3.4 Movie Recommender System Based on Collaborative Filtering Using Apache Spark

Collaborative filtering (CF) with alternating least squares (ALS) algorithm is the most imperative techniques which are used for building a movie recommendation engine. The ALS algorithm is one of the models of matrix factorization related CF which is considered as the values in the item list of user matrix. As there is a need to perform analysis on the ALS algorithm by selecting different parameters which can eventually help in building efficient movie recommender engine. In this work, we propose a movie recommender system based on ALS using Apache Spark. This research focuses on the selection of parameters of ALS algorithms that can affect the performance of a building robust RS. From the results, a conclusion is drawn according to the selection of parameters of ALS algorithms which can affect the performance of building of a movie recommender engine. The model evaluation is done using different metrics such as execution time, root mean squared error (RMSE) of rating prediction, and rank in which the best model was trained. Two best cases are chosen based on best parameters selection from experimental results which can lead to building good prediction rating for a movie recommender.

3.4.1 Proposed Model

This section provides the idea of the proposed system. The proposed system is a movie recommender system based on ALS using Apache Spark. The novelty of this work is based on the selection of parameters of ALS algorithms that can affect the performance of building of a movie recommender system.

In this work, we apply user's ratings from the datasets the popular website like IMDB, Rotten Tomatoes, MovieLen, and Time Movie Ratings. This dataset is available in many formats such as CSV file, text file, and databases. We can either stream the data live from the websites or download and store them on our local file system or HDFS. Spark streaming is used to stream real-time data from the various source like Twitter, the stock market, and geographical system and perform powerful analytic to businesses. It used for processing real-time streaming data. We use collaborative filtering (CF) to predict the ratings of users for particular movies based on their ratings for other movies. Then collaborate this with another user's rating for that particular movie. We train the ALS algorithm using MovieLen data and get the results from the machine learning model. We use spark SQL's data frame, dataset, and SQL service to store the data. The result of the machine learning model is stored in RDBMS so that the web application can display the recommendation to a particular use. The results of the movie recommendation system are stored in our local drive. We store the recommendation movies along with the ratings in a text file and CSV file formats. We prefer storing the result into an RDBMS system so as to access it directly from the web application and display recommendation and top movies as shown in Fig 11.

Figure 11: Proposed Movie Recommendation System using CF with ALS

3.4.2 Experimental Results Analysis and Discussion

In this work, Apache Spark is used to demonstrate an efficient parallel implementation of a Collaborative Filtering method using alternating least squares (ALS). ALS is used for dimensionality reduction purpose which helps in overcoming the limitations of Col-

Movie Recommendation System using CF with ALS

Input: MovieLens Dataset
Output: Top Recommended Movies.
Procedures:
Procedure 1:Parsing and loading datasets
Procedure 2: Recognize the user as new or regular.
If new user goto **Procedure** 5
Procedure 3: Load training and test data into the table (userId, movieId, rating)
def parse_the_rating(line):
x = line.split()
return (int (x [0]), int (x [1]), float (x [2]))
training = sc.TrainingFile("__").map(parse_the_Rating).cache()
test = sc.Testfile("__").map(parse_the_Rating)
Procedure 4: Train the recommender model.
New_model= ALS.train (rank, train, iteration)
Procedure 5:Create predictions on (user, movie) pairs from the test data
Predict = New_model.predictAll (test.map(lambda x: (x[0], x[1]))
Procedure6: Adding new user ratings
Procedure 7: Display top N recommended movies.
Procedure 8: Save the New_model

Figure 12: Proposed System Steps

laborative Filtering such as data sparsity and scalability. The challenges of data sparsity are appearing in numerous situations, specifically, another problem, when a new an item or user has just added to the system, it is difficult to find similar ones since there is no sufficient information, this problem is called cold start problem [96] . When selecting the ALS algorithm as a part of building the proposed movie recommender system, there is basic parameter through them can determine the best rating of users for given movies. These parameters are Rank, Iterations, and Lambda. The contribution of this paper is to study and determine the selection of parameters that affect the performance of ALS model in building a movie recommender system because from literature study, it's found that little research work focused on the study of the selection of ALS's parameters that can affect its performance in building a movie recommender engine using Apache Spark. The parameters, lambda, and iterations are used in order to control and adjust the predicting capability of matrix factorization which is depending on ALS technique which in turn affect the evaluation of movie RS. The iterations and lambda parameters are used as follows; Lambda which specifies the regularization parameter in ALS, Iterations which proposed model should run the specified number of iterations. The ALS algorithm achieves its optimal solution between **5** and **20** iterations.

The parameters lambda and iteration in ALS model are used with different thresholds to realize the effects of matrix factorization performance on the performance of recommendation results and thus take the most appropriate parameters for the following test setups. Table 9 to Table 11 show the performance of movie recommendation engine based on ALS under different values of lambda and iteration. Table 9 illustrates the execution of time with the changes of lambda with iterations parameters of ALS model, while Table 10 the Rank of best-trained model with the changes of lambda with iterations parameters of ALS algorithm, finally Table 11 Indicatesthe RMSE with the changes of lambda with iterations parameters of ALS model. The results presented in Table 9 indicates that when lambda is set to **0.6** and iteration set is **10**, the time value is minimum which is **1.41323** seconds, and rank value is **8** as shown in Table 10. Moreover, the RMSE register for this

rating is **1.07424** as indicated in Table 11. On the other hand, as it is indicated in Table 9 when lambda is set to **0.2** and iteration is **15**, running time becomes **1.463743** and Rank are 12 for this item as shown in Table 10. The RMSE value for this item is the minimum, which is **0.9167**, as presented in Table 11.

As mentioned above, the analysis for movie recommendation system is done using three quality metrics which are RMSE, Time and Rank. Using these three metrics, two cases are achieved as shown in Table 12, case 1 with high time and low RMSE rate while the case 2 with low time and high RMSE rate. According to results in Table 12, the prediction for Top 25 movies is shown in Fig 13 and Fig 14. In general, the lowest value of the RMSE is considered the best case for prediction in building recommendation system. Therefore, we will adopt the second case because the value of the RMSE is smaller compared to the value in the first case as well as adopt the second case as the best case because there is no significant difference in the amount of time execution between the two cases. Now we can get the top recommended movies by using the second case. Finally, we concluded that from these results the best case is the second case which has the best value for RMSE, which can useful for building recommendation engines for predicting the top 25 ranked movies.

Table 9: Time of matrix factorization using lambda and iteration parameters

Lamda	Iterations				
	5	10	15	20	25
0.1	1.489	1.454	1.473	1.469	1.512
0.2	1.485	1.437	**1.464**	1.438	1.514
0.3	1.472	1.481	1.4671	1.441	1.494
0.4	1.658	1.486	1.476	1.495	1.473
0.5	1.431	1.492	1.468	1.478	1.528
0.6	1.615	**1.413**	1.442	1.459	1.480
0.7	1.443	1.475	1.471	1.446	1.543
0.8	1.554	1.470	1.459	1.449	1.527
0.9	1.491	1.478	1.482	1.471	1.446

Table 10: Rank of matrix factorization using lambda and iteration parameters

Lamda	Itrations				
	5	10	15	20	25
0.1	4	12	4	4	4
0.2	8	12	**12**	12	12
0.3	4	8	8	8	8
0.4	4	8	8	8	8
0.5	8	8	8	8	8
0.6	8	**8**	8	8	12
0.7	12	12	4	4	12
0.8	12	12	4	4	4
0.9	12	4	4	4	4

Table 11: RMSE of matrix factorization using lambda and iteration parameters

Lamda	Itrations				
	5	10	15	20	25
0.1	0.947	0.942	0.940	0.938	0.938
0.2	0.919	0.917	**0.9167**	0.917	0.917
0.3	0.941	0.941	0.941	0.941	0.941
0.4	0.975	0.980	0.980	0.981	0.981
0.5	1.018	1.024	1.024	1.024	1.024
0.6	1.069	**1.074**	1.074	1.074	1.074
0.7	1.127	1.130	1.131	1.131	1.131
0.8	1.192	1.193	1.193	1.193	1.193
0.9	1.261	1.261	1.261	1.261	1.261

Table 12: Two cases for selecting parameters for ALS

Metrics	Case	
	Case1	Case2
Time	1.41323	**1.4637**
Rank	8	**12**
RMSE	1.07422	**0.9167**

My 25 highest rated movies as predicted (for movies with more than 75 reviews):

movieId	userId	prediction	movieId	count	average	title
159817	0	3.8603234	159817	193	4.450777202072539	Planet Earth (2006)
318	0	3.8078098	318	84455	4.43308862707951	Shawshank Redempt...
160718	0	3.797955	160718	88	4.232954545454546	Piper (2016)
134849	0	3.745416	134849	120	4.041666666666667	Duck Amuck (1953)
858	0	3.7449322	858	53547	4.343623358918333	Godfather, The (1...
26048	0	3.7382393	26048	76	3.9539473684210527	Human Condition I...
50	0	3.7262065	50	56348	4.308635621494996	Usual Suspects, T...
100044	0	3.722707	100044	97	4.15979381443299	Human Planet (2011)
142115	0	3.7207592	142115	150	4.31	The Blue Planet (...
2019	0	3.7023807	2019	13394	4.256196804539346	Seven Samurai (Sh...
94466	0	3.7008467	94466	2477	4.2426322163907955	Black Mirror (2011)
162376	0	3.6991944	162376	703	4.312944523470839	Stranger Things
77658	0	3.6968207	77658	1740	4.1818965517241375	Cosmos (1980)
527	0	3.6902485	527	63889	4.2759629983252205	Schindler's List ...
904	0	3.6900668	904	20443	4.238590226483393	Rear Window (1954)
26082	0	3.6886785	26082	542	4.100553505535055	Harakiri (Seppuku...
1178	0	3.6881678	1178	4064	4.20435531496063	Paths of Glory (1...
922	0	3.6870744	922	7606	4.2058900867736	Sunset Blvd. (a.k...
7926	0	3.6856618	7926	703	4.105263157894737	High and Low (Ten...
1212	0	3.6842163	1212	7412	4.214854290339989	Third Man, The (1...
6669	0	3.6837091	6669	1340	4.1093283582089555	Ikiru (1952)
3030	0	3.6817052	3030	3988	4.186434302908726	Yojimbo (1961)
127052	0	3.675985	127052	160	4.1	Operation 'Y' & O...
3435	0	3.672548	3435	5421	4.203836930455635	Double Indemnity ...
2920	0	3.6651812	2920	1115	4.1246636771300444	Children of Parad...

only showing top 25 rows

Figure 13: Prediction of top 25 movies for case 1

My 25 highest rated movies as predicted (for movies with more than 75 reviews):

movieId	userId	prediction	movieId	count	average	title
159817	0	4.6047864	159817	193	4.450777202072539	Planet Earth (2006)
318	0	4.5063915	318	84455	4.43308862707951	Shawshank Redempt...
160718	0	4.466501	160718	88	4.232954545454546	Piper (2016)
134849	0	4.4416428	134849	120	4.041666666666667	Duck Amuck (1953)
858	0	4.429425	858	53547	4.343623358918333	Godfather, The (1...
527	0	4.413663	527	63889	4.2759629983252205	Schindler's List ...
100044	0	4.395683	100044	97	4.15979381443299	Human Planet (2011)
100553	0	4.3911767	100553	239	4.062761506276151	Frozen Planet (2011)
50	0	4.387128	50	56348	4.308635621494996	Usual Suspects, T...
912	0	4.383373	912	29001	4.220164821902693	Casablanca (1942)
904	0	4.382594	904	20443	4.238590226483393	Rear Window (1954)
77658	0	4.3616266	77658	1740	4.1818965517241375	Cosmos (1980)
77177	0	4.3589344	77177	91	3.791208791208791	Wild China (2008)
1203	0	4.3545713	1203	15938	4.224777261889823	12 Angry Men (1957)
2019	0	4.3540235	2019	13394	4.256196804539346	Seven Samurai (Sh...
108583	0	4.3512716	108583	921	4.074918566775244	Fawlty Towers (19...
127052	0	4.349377	127052	160	4.1	Operation 'Y' & O...
142115	0	4.3470507	142115	150	4.31	The Blue Planet (...
44555	0	4.343796	44555	8241	4.1996723698580265	Lives of Others, ...
1207	0	4.3423305	1207	16738	4.162474608674872	To Kill a Mocking...
908	0	4.339527	908	18467	4.208669518600748	North by Northwes...
93040	0	4.3393526	93040	380	4.044736842105263	Civil War, The (1...
94466	0	4.3387938	94466	2477	4.2426322163907955	Black Mirror (2011)
82143	0	4.3382115	82143	295	3.906779661016949	Alone in the Wild...
162376	0	4.3322854	162376	703	4.312944523470839	Stranger Things

only showing top 25 rows

Figure 14: Prediction of top 25 movies for case 2

4 Conclusion and Future works

In this research work, we have provided a critical review on challenges, solutions, conventional and newly emerged of collabrative filtering algorithms recommender systems.

To overcome the sparsity and scalability of collabrative filtering algorithms, we have proposed a a hybrid recommendation model namely ΓUICF. The ΓUICF model combines UCF and ICF linear regression models to modeling unknown user-item preferences. The ΓUICF model was implemented on two different real-world datasets namely 100k MovieLens and M1 MovieLens datasets. Our results demonstrate that the accuracy of the proposed recommendation model has got the significant performance and minimum error compared to existing CF models, which is showing the outperformance of our approach compared to the existing collaborative filtering (CF) models. The proposed model solved the limitations of CF algorithms, however, there is some shortage of CF models when dealing with real-world applications. It does not effectively address the well-known cold start problem, that is when a new user or new item enters the system. There is difficulty to deal with sparse data, hence it's hard to find users that have scored the same items. It suffers when new users or items that don't have any ratings enter the system. It tends to recommend popular items.

To solve the cold start and sparsity problems, we proposed a new recommendation model based on Non-IID that integrates two parts, namely explicit user-item interaction and implicit user-item interactions. The explicit interactions are divided into two models: Intra-coupled interactions within the features of users and items and Inter-coupled interactions learning between different features of users and items, with both forms of learning using convolution neural networks. We learned the implicit interactions using generalized matrix factorization with bias (GMFB). The explicit and implicit learning interactions were then combined under a framework known as the 'IntegrateCF'. Extensive experiments on two large real-world datasets have shown that the proposed model outperforms existing methods on the MovieLens datasets. For the Tafeng dataset, our model was high between 1-4 epochs and declined after that. We presented a movie recommender system based on collaborative filtering using Apache Spark. As a result, the selection of parameters of (Alternating Least Squares) ALS algorithms can affect the performance of the building of a movie recommendation engine. System evaluation is done using various metrics such as execution time, RMSE of rating prediction, and rank in which the best model was trained. Two best cases are chosen based on best parameter selection from experimental results, which can lead to building a god prediction rating for a movie recommender engine. From these cases, the lowest value of the RMSE is considered the best case for prediction in building a movie recommendation system. Therefore, the second case is recommended to be used since the value of the RMSE is smaller compared to the value in the first case, as well as adopt the second case as the best case, because there is no significant difference in the amount of time spent on execution between the two cases. Finally, we concluded that from these results, the best case is the second case, which has the best value for RMSE and can be useful for building recommendation engines for predicting the top 25 ranked movies. In the future work, we plan to develop and improve a new loss function because of the shortcomings of the recommender system algorithm based on the ALS model, based on the parameter of the best case, which has the best value for RMSE using Apache Spark.

In future work, we intend to propose a novel recommendation model using the deep

learning method, add another feature relating to user and item characteristics, and use an attention technique to improve the recommendation system's accuracy. We will also fine-tune our work to make it applicable online.

5 Publications

- Aljunid, Mohammed Fadhel, and Manjaiah Doddaghatta Huchaiah. **Efficient Hybrid Recommendation model Based on Collaborative Filtering Recommender Systems.** upcoming publication in journal **CAAI Transactions on Intelligence Technology.**
- Aljunid, Mohammed Fadhel, and Manjaiah Doddaghatta Huchaiah. **Multi-model deep learning approach for collaborative filtering recommendation system.** CAAI Transactions on Intelligence Technology 5.4 (2020): 268-275. `https://doi.org/10.1049/trit.2020.0031`
- Aljunid, Mohammed Fadhel, and Manjaiah Dh. **An Efficient Deep Learning Approach for Collaborative Filtering Recommender System.** Procedia Computer Science 171 (2020): 829-836.`ttps://doi.org/10.1016/j.procs.2020.04.090`
- Aljunid M.F., Manjaiah D.H. (2019) **Movie Recommender System Based on Collaborative Filtering Using Apache Spark.** In: Balas V., Sharma N., Chakrabarti A. (eds) Data Management, Analytics and Innovation. Advances in Intelligent Systems and Computing, vol 839. Springer, Singapore. `https://doi.org/10.1007/978-981-13-1274-8_22`
- Aljunid, Mohammed Fadhel, and D. H. Manjaiah. **An improved ALS recommendation model based on apache spark." International Conference on Soft Computing Systems.** Springer, Singapore, 2018. `https://doi.org/10.1007/978-981-13-1936-5_33`
- FadhelAljunid, Mohammed, and D. H. Manjaiah. **A survey on recommendation systems for social media using big data analytics.** International Journal of Latest Trends in Engineering and Technology (2017): 48-58. `https://www.ijltet.org/journal_details.php?id=922&j_id=4043`

References

[1] Mehmet Aktukmak, Yasin Yilmaz, and Ismail Uysal. A probabilistic framework to incorporate mixed-data type features: Matrix factorization with multimodal side information. *Neurocomputing*, 367:164–175, 2019.

[2] Nuha Alghamdi and Fatmah Assiri. Solving the cold-start problem in recommender systems using contextual information in arabic from calendars. *Arabian Journal for Science and Engineering*, 45(12):10939–10947, 2020.

[3] Yilmaz Ar. An initialization method for the latent vectors in probabilistic matrix factorization for sparse datasets. *Evolutionary Intelligence*, pages 1–13, 2019.

[4] Andrey Babkin. Incorporating side information into robust matrix factorization with bayesian quantile regression. *Statistics & Probability Letters*, 165:108847, 2020.

[5] Payam Bahrani, Behrouz Minaei-Bidgoli, Hamid Parvin, Mitra Mirzarezaee, Ahmad Keshavarz, and Hamid Alinejad-Rokny. User and item profile expansion for dealing with cold start problem. *Journal of Intelligent & Fuzzy Systems*, (Preprint):1–13, 2020.

[6] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[7] Saumya Bansal and Niyati Baliyan. Bi-mars: A bi-clustering based memetic algorithm for recommender systems. *Applied Soft Computing*, 97:106785, 2020.

[8] Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kaleli. A review on deep learning for recommender systems: challenges and remedies. *Artificial Intelligence Review*, 52(1):1–37, 2019.

[9] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Jesús Bernal. Generalization of recommender systems: Collaborative filtering extended to groups of users and restricted to groups of items. *Expert systems with Applications*, 39(1):172–186, 2012.

[10] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *arXiv preprint arXiv:1301.7363*, 2013.

[11] Fidel Cacheda, Víctor Carneiro, Diego Fernández, and Vreixo Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web (TWEB)*, 5(1):1–33, 2011.

[12] Longbing Cao. Coupling learning of complex interactions. *Information Processing & Management*, 51(2):167–186, 2015.

[13] Guangyong Chen, Fengyuan Zhu, and Pheng Ann Heng. Large-scale bayesian probabilistic matrix factorization with memo-free distributed variational inference. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(3):1–24, 2018.

[14] Lei Chen, Zhiang Wu, Jie Cao, Guixiang Zhu, and Yong Ge. Travel recommendation via fusing multi-auxiliary information into matrix factorization. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(2):1–24, 2020.

[15] Meng-Hui Chen, Chin-Hung Teng, and Pei-Chann Chang. Applying artificial immune systems to collaborative filtering for movie recommendation. *Advanced Engineering Informatics*, 29(4):830–839, 2015.

[16] Shulong Chen and Yuxing Peng. Matrix factorization for recommendation with explicit and implicit feedback. *Knowledge-Based Systems*, 158:109–117, 2018.

[17] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[18] Zhihua Cui, Xianghua Xu, Fei Xue, Xingjuan Cai, Yang Cao, Wensheng Zhang, and Jinjun Chen. Personalized recommendation system based on collaborative filtering for iot scenarios. *IEEE Transactions on Services Computing*, 2020.

[19] Mehri Davtalab and Ali Asghar Alesheikh. A poi recommendation approach integrating social spatio-temporal information into probabilistic matrix factorization. *Knowledge and Information Systems*, pages 1–21, 2020.

[20] Marco De Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. Semantics-aware content-based recommender systems. In *Recommender systems handbook*, pages 119–159. Springer, 2015.

[21] Jiangzhou Deng, Junpeng Guo, and Yong Wang. A novel k-medoids clustering recommendation algorithm based on probability distribution for collaborative filtering. *Knowledge-Based Systems*, 175:96–106, 2019.

[22] Zhi-Hong Deng, Ling Huang, Chang-Dong Wang, Jian-Huang Lai, and S Yu Philip. Deepcf: A unified framework of representation learning and matching function learning in recommender system. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 61–68, 2019.

[23] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 107–144. Springer, 2011.

[24] Mohammed FadhelAljunid and DH Manjaiah. A survey on recommendation systems for social media using big data analytics. *International Journal of Latest Trends in Engineering and Technology*, pages 48–58, 2017.

[25] Chenjiao Feng, Jiye Liang, Peng Song, and Zhiqiang Wang. A fusion collaborative filtering method for sparse data in recommender systems. *Information Sciences*, 521:365–379, 2020.

[26] Simon Funk. Netflix update: Try this at home, 2006.

[27] Achraf Gazdar and Lotfi Hidri. A new similarity measure for collaborative filtering based recommender systems. *Knowledge-Based Systems*, 188:105058, 2020.

[28] Marco Gori, Augusto Pucci, V Roma, and I Siena. Itemrank: A random-walk based scoring algorithm for recommender engines. In *IJCAI*, volume 7, pages 2766–2771, 2007.

[29] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

[30] Zahra Yusefi Hafshejani, Marjan Kaedi, and Afsaneh Fatemi. Improving sparsity and new user problems in collaborative filtering by clustering the personality factors. *Electronic Commerce Research*, 18(4):813–836, 2018.

[31] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

[32] J Herce-Zelaya, C Porcel, J Bernabé-Moreno, A Tejeda-Lorente, and E Herrera-Viedma. New technique to alleviate the cold start problem in recommender systems using information from social media and random decision forests. *Information Sciences*, 536:156–170, 2020.

[33] Antonio Hernando, Jesús Bobadilla, and Fernando Ortega. A non negative matrix factorization for collaborative filtering recommender systems based on a bayesian probabilistic model. *Knowledge-Based Systems*, 97:188–202, 2016.

[34] Yassine Himeur, Abdullah Alsalemi, Ayman Al-Kababji, Faycal Bensaali, Abbes Amira, Christos Sardianos, George Dimitrakopoulos, and Iraklis Varlamis. A survey of recommender systems for energy efficiency in buildings: Principles, challenges and prospects. *Information Fusion*, 72:1–21, 2021.

[35] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. Collaborative metric learning. In *Proceedings of the 26th international conference on world wide web*, pages 193–201, 2017.

[36] Nouhaila Idrissi and Ahmed Zellou. A systematic literature review of sparsity issues in recommender systems. *Social Network Analysis and Mining*, 10(1):1–23, 2020.

[37] FO Isinkaye, YO Folajimi, and Bolande Adefowoke Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian informatics journal*, 16(3):261–273, 2015.

[38] Dietmar Jannach, Zeynep Karakaya, and Fatih Gedikli. Accuracy improvements for multi-criteria recommender systems. In *Proceedings of the 13th ACM conference on electronic commerce*, pages 674–689, 2012.

[39] Dazhi Jiang, Zhichao Liu, Lin Zheng, and Jianwei Chen. Factorization meets neural networks: A scalable and efficient recommender for solving the new user problem. *IEEE Access*, 8:18350–18361, 2020.

[40] Surya Kant and Tripti Mahara. Merging user and item based collaborative filtering to alleviate data sparsity. *International Journal of System Assurance Engineering and Management*, 9(1):173–179, 2018.

[41] Surya Kant and Tripti Mahara. Nearest biclusters collaborative filtering framework with fusion. *Journal of Computational Science*, 25:204–212, 2018.

[42] Surya Kant, Tripti Mahara, Vinay Kumar Jain, Deepak Kumar Jain, and Arun Kumar Sangaiah. Leaderrank based k-means clustering initialization method for collaborative filtering. *Computers & Electrical Engineering*, 69:598–609, 2018.

[43] Kyung Soo Kim, Doo Soo Chang, and Yong Suk Choi. Boosting memory-based collaborative filtering using content-metadata. *Symmetry*, 11(4):561, 2019.

[44] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[45] R Kiran, Pradeep Kumar, and Bharat Bhasker. Dnnrec: A novel deep learning based hybrid recommender system. *Expert Systems with Applications*, 144:113054, 2020.

[46] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer, 2015.

[47] Georgia Koutrika, Benjamin Bercovitz, and H FlexRecs Garcia-Molina. Expressing and combining flexible recommendations. In *Proceedings of the 35th SIGMOD International Conference on Management of Data (SIGMOD'09), Providence, RI, USA*, volume 29.

[48] Vikas Kumar, Arun K Pujari, Sandeep Kumar Sahu, Venkateswara Rao Kagita, and Vineet Padmanabhan. Collaborative filtering using multiple binary maximum margin matrix factorizations. *Information Sciences*, 380:1–11, 2017.

[49] Soojung Lee. Using entropy for similarity measures in collaborative filtering. *Journal of Ambient Intelligence and Humanized Computing*, 11(1):363–374, 2020.

[50] Cane Wing-ki Leung, Stephen Chi-fai Chan, and Fu-lai Chung. An empirical study of a cross-level association rule mining approach to cold-start recommendations. *Knowledge-Based Systems*, 21(7):515–529, 2008.

[51] Haifeng Liu, Zheng Hu, Ahmad Mian, Hui Tian, and Xuzhen Zhu. A new user similarity model to improve the accuracy of collaborative filtering. *Knowledge-Based Systems*, 56:156–166, 2014.

[52] Jian Liu and Youling Chen. A personalized clustering-based and reliable trust-aware qos prediction approach for cloud service recommendation in cloud manufacturing. *Knowledge-Based Systems*, 174:43–56, 2019.

[53] Jiangzhou Liu and Li Duan. A survey on knowledge graph-based recommender systems. In *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, volume 5, pages 2450–2453. IEEE, 2021.

[54] Qi Liu, Enhong Chen, Hui Xiong, Chris HQ Ding, and Jian Chen. Enhancing collaborative filtering by user interest expansion via personalized ranking. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(1):218–233, 2011.

[55] R Logesh, V Subramaniyaswamy, D Malathi, N Sivaramakrishnan, and V Vijayakumar. Enhancing recommendation stability of collaborative filtering recommender system through bio-inspired clustering ensemble method. *Neural Computing and Applications*, 32(7):2141–2164, 2020.

[56] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[57] Supriyo Mandal and Abyayananda Maiti. Explicit feedback meet with implicit feedback in gpmf: a generalized probabilistic matrix factorization model for recommendation. *Applied Intelligence*, pages 1–24, 2020.

[58] Zhiyu Min and Dahua Lin. Probabilistic ensemble of collaborative filters. *arXiv preprint arXiv:1808.03298*, 2018.

[59] Maryam Khanian Najafabadi, Mohd Naz'ri Mahrin, Suriayati Chuprat, and Haslina Md Sarkan. Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data. *Computers in Human Behavior*, 67:113–128, 2017.

[60] Senthilselvan Natarajan, Subramaniyaswamy Vairavasundaram, Sivaramakrishnan Natarajan, and Amir H Gandomi. Resolving data sparsity and cold start problem in collaborative filtering recommender system using linked open data. *Expert Systems with Applications*, 149:113248, 2020.

[61] Behzad Soleimani Neysiani, Nasim Soltani, Reza Mofidi, and Mohammad Hossein Nadimi-Shahraki. Improve performance of association rule-based collaborative filtering recommendation systems using genetic algorithm. *Int. J. Inf Technol. Comput. Sci*, 2:48–55, 2019.

[62] Patricia Ortal and Masato Edahiro. Switching hybrid method based on user similarity and global statistics for collaborative filtering. *IEEE Access*, 8:213401–213415, 2020.

[63] Sanjaya Kumar Panda, Sourav Kumar Bhoi, and Munesh Singh. A collaborative filtering recommendation algorithm based on normalization approach. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–23, 2020.

[64] Manh Cuong Pham, Yiwei Cao, Ralf Klamma, and Matthias Jarke. A clustering approach for collaborative filtering recommendation using social network analysis. *J. UCS*, 17(4):583–604, 2011.

[65] Abinash Pujahari and Dilip Singh Sisodia. Pair-wise preference relation based probabilistic matrix factorization for collaborative filtering in recommender system. *Knowledge-Based Systems*, page 105798, 2020.

[66] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[67] Fatemeh Rezaimehr and Chitra Dadkhah. A survey of attack detection approaches in collaborative filtering recommender systems. *Artificial Intelligence Review*, 54(3):2011–2066, 2021.

[68] Masoumeh Riyahi and Mohammad Karim Sohrabi. Providing effective recommendations in discussion groups using a new hybrid recommender system based on implicit ratings and semantic similarity. *Electronic Commerce Research and Applications*, 40:100938, 2020.

[69] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887, 2008.

[70] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. Item-based collaborative filtering recommendation algorithms. *Www*, 1:285–295, 2001.

[71] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[72] Monika Singh. Scalability and sparsity issues in recommender datasets: a survey. *Knowledge and Information Systems*, 62(1):1–43, 2020.

[73] Monika Singh and Monica Mehrotra. Impact of biclustering on the performance of biclustering based collaborative filtering. *Expert Systems With Applications*, 113:443–456, 2018.

[74] Pradeep Kumar Singh, Showmik Setta, Pijush Kanti Dutta Pramanik, and Prasenjit Choudhury. Improving the accuracy of collaborative filtering-based recommendations by considering the temporal variance of top-n neighbors. In *International Conference on Innovative Computing and Communications*, pages 1–10. Springer, 2020.

[75] Nathan Srebro, Jason Rennie, and Tommi Jaakkola. Maximum-margin matrix factorization. *Advances in neural information processing systems*, 17:1329–1336, 2004.

[76] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[77] Chuan Sun, Hui Li, Xiuhua Li, Junhao Wen, Qingyu Xiong, and Wei Zhou. Convergence of recommender systems and edge computing: A comprehensive survey. *IEEE Access*, 8:47118–47132, 2020.

[78] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Latent relational metric learning via memory-based attention for collaborative ranking. In *Proceedings of the 2018 World Wide Web Conference*, pages 729–739, 2018.

[79] Cong Tran, Jang-Young Kim, Won-Yong Shin, and Sang-Wook Kim. Clustering-based collaborative filtering using an incentivized/penalized user model. *IEEE Access*, 7:62115–62125, 2019.

[80] Daniel Valcarce, Alfonso Landin, Javier Parapar, and Álvaro Barreiro. Collaborative filtering embeddings for memory-based recommender systems. *Engineering Applications of Artificial Intelligence*, 85:347–356, 2019.

[81] Daniel Valcarce, Javier Parapar, and Álvaro Barreiro. Finding and analysing good neighbourhoods to improve collaborative filtering. *Knowledge-Based Systems*, 159:193–202, 2018.

[82] Priscila Valdiviezo-Diaz, Fernando Ortega, Eduardo Cobos, and Raúl Lara-Cabrera. A collaborative filtering approach based on naïve bayes classifier. *IEEE Access*, 7:108581–108592, 2019.

[83] Bogdan Walek and Vladimir Fojtik. A hybrid recommender system for recommending relevant movies using an expert system. *Expert Systems with Applications*, 158:113452, 2020.

[84] Dawei Wang, Yuehwern Yih, and Mario Ventresca. Improving neighbor-based collaborative filtering by using a hybrid similarity measurement. *Expert Systems with Applications*, 160:113651, 2020.

[85] Jing Wang and Jian Yin. Combining user-based and item-based collaborative filtering techniques to improve recommendation diversity. In *2013 6th International Conference on Biomedical Engineering and Informatics*, pages 661–665. IEEE, 2013.

[86] Mohammed Wasid and Rashid Ali. Fuzzy side information clustering-based framework for effective recommendations. *Computing and Informatics*, 38(3):597–620, 2019.

[87] Akihiro Yamashita, Hidenori Kawamura, and Keiji Suzuki. Adaptive fusion method for user-based and item-based collaborative filtering. *Advances in Complex Systems*, 14(02):133–149, 2011.

[88] Hafed Zarzour, Ziad Al-Sharif, Mahmoud Al-Ayyoub, and Yaser Jararweh. A new collaborative filtering recommendation algorithm based on dimensionality reduction and clustering techniques. In *2018 9th International Conference on Information and Communication Systems (ICICS)*, pages 102–106. IEEE, 2018.

[89] Fuguo Zhang, Shumei Qi, Qihua Liu, Mingsong Mao, and An Zeng. Alleviating the data sparsity problem of recommender systems by clustering nodes in bipartite networks. *Expert Systems with Applications*, 149:113346, 2020.

[90] Guijuan Zhang, Yang Liu, and Xiaoning Jin. A survey of autoencoder-based recommender systems. *Frontiers of Computer Science*, 14(2):430–450, 2020.

[91] Lingxiao Zhang, Jiangpeng Yan, Yujiu Yang, and Li Xiu. Match4rec: A novel recommendation algorithm based on bidirectional encoder representation with the matching task. In *International Conference on Neural Information Processing*, pages 491–503. Springer, 2020.

[92] Quangui Zhang, Longbing Cao, Chengzhang Zhu, Zhiqiang Li, and Jinguang Sun. Coupledcf: Learning explicit and implicit user-item couplings in recommendation for deep collaborative filtering. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

[93] Quangui Zhang, Li Wang, Xiangfu Meng, Keda Xu, and Jiayan Hu. A generic framework for learning explicit and implicit user-item couplings in recommendation. *IEEE Access*, 7:123944–123958, 2019.

[94] Yuhang Zhang, Jun Wang, and Jie Luo. Knowledge graph embedding based collaborative filtering. *IEEE Access*, 8:134553–134562, 2020.

[95] Zhipeng Zhang, Yao Zhang, and Yonggong Ren. Employing neighborhood reduction for alleviating sparsity and cold start problems in user-based collaborative filtering. *Information Retrieval Journal*, 23(4):449–472, 2020.

[96] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *International conference on algorithmic applications in management*, pages 337–348. Springer, 2008.

[97] Jun Zhu, Lixin Han, Zhinan Gou, and Xiaofeng Yuan. A fuzzy clustering-based denoising model for evaluating uncertainty in collaborative filtering recommender systems. *Journal of the Association for Information Science and Technology*, 69(9):1109–1121, 2018.

Research Scholar:

Mohammed Ahmed Mohammed Fadhel Aljunid,
Research Scholar,
Department of Computer Science,
Mangalore University.

Research Supervisor:

Dr. Manjaiah D.H,
Professor,
Department of Computer Science,
Mangalore University.